Scalable Methods for Multivariate Time Series Classification

by

Bhaskar Dhariyal

This thesis is submitted to University College Dublin in fulfilment of the requirements for the degree of Doctor of Philosophy

School of Computer Science

Head of School: Assoc. Prof. Neil Hurley

Principal Supervisor: Assoc. Prof. Georgiana Ifrim

Doctoral Studies Panel Members:

Prof. Mark Keane
Assoc. Prof. Brian MacNamee

January 2024

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

Acronym	Explanation
TSC	Time Series Classification
SOTA	State-of-the-Art
MTSC	Multivariate Time Series Classification
UTSC	Univariate Time Series Classification
UCR/UEA	UCR/UEA Time Series Classification Archive
SAX	Symbolic Aggregate approXimation
SFA	Symbolic Fourier approXimation
PAA	Piecewise Aggregate Approximation
DTW	Dynamic Time Warping
WEASEL-MUSE	Word ExtrAction for time SEries cLassification- MUltivariate Symbolic Extension
MrSEQL	Multi-resolution SeQuence Learner
ROCKET	RandOm Convolutional KErnel Transform
MiniRocket	MINImally RandOm Convolutional KErnel Transform

ABSTRACT

The availability of low-tech sensors has significantly impacted the goals of industry and academia. These sensors are now used in various health, agriculture, and finance applications. Their usage has led to the proliferation of temporal data, presenting challenges and opportunities. Temporal data is a series of real values collected over time. The changes in the temporal data are used to improve our understanding of how systems work and can also be used to make predictions. For example, data recorded from an athlete's run spanning over a few minutes could help identify a potential injury. The ease of availability of temporal data has opened up new opportunities for research and development. However, it also presents unique challenges. One challenge is that temporal data can be vast and complex, making storing, processing, and analysing difficult.

The advancement in data capture technology and machine learning approaches have enabled us to tackle some of these challenges. This thesis focuses on studying and developing machine learning methods for analysing temporal data. In particular, it focuses on a type of temporal data called multivariate time series and a specific machine learning task called time series classification. Considering the above example of an athlete's run, data captured from the run is multivariate as it involves collecting motion data from multiple sources like arms, legs, and waist. Each source, in this case, a body part, is known as a channel. The task of predicting a discrete label, for example, injury type, on this temporal data is called multivariate time series classification. The multiple channels in a multivariate time series classification task can result in more computation time for different classifiers as the data increases proportionately to the number of channels. Therefore, reducing these delays is sometimes as crucial as the accuracy of classifiers. These delays can be broadly mapped to the scalability of classifiers. In this thesis, we examine various aspects of scalability for different types of time series datasets and time series classification algorithms and propose solutions to address them.

In the first contribution, we examine state-of-the-art multivariate time series classifiers on standard benchmark datasets and identify the issues affecting the scalability of these classifiers. We identify three main problems affecting scalability: the number of data points, the number of channels, and the length of the time series. To address the problem of the time series length, we propose a method to extract statistical properties of the time series. We use statistical properties of time series, which are fast to compute

and easier to implement. When combined with techniques such as windowing and smoothing, we observe that they enhance the performance of the studied models.

In the second contribution, we address the challenge of the increasing computation time caused by the number of channels in multivariate time series classification. When there are more channels, the classifiers need to process more data. Additionally, longer time series can further amplify this scalability issue. To address this problem, we proposed a strategy for efficient channel selection. This strategy reduces the amount of data that needs to be processed by the classifiers, which leads to significant speedups. We show how the distance between the centroid of two classes can be used to distinguish between useful and not-useful channels. We also illustrate the performance of state-of-the-art classifiers with and without channel selection on the real-life Military Press dataset. Our results show that our channel selection strategy can significantly reduce computation time and memory requirements while maintaining classification accuracy. This makes it a valuable tool for improving the scalability of multivariate time series classification.

In the third contribution, we address the problem affecting the centroid as a class representation. The problem is that the centroid is susceptible to noise and outliers. To address this issue, we propose two new class representations, namely, the median and the median absolute deviation (MAD), which are more robust to noise and outliers. We compare the performance of these class prototypes with the centroid class prototype. We extensively evaluate these class prototypes, benchmarking them on the popular datasets established in the multivariate time series classification community. We also compare them on synthetic datasets by generating different data of different characteristics, such as noise and impact on the increasing number of channels, while varying the number of useful channels.

In this thesis's fourth and last contribution, we examine the latest developments in time series classification. We compare more recent state-of-the-art time series models with simple tabular models on multivariate time series in terms of accuracy and computation time. Many existing evaluation studies do not consider tabular methods as baselines, and the performance is reported in aggregate, for example, as average rank or average accuracy. This can hide that some methods are better on specific domains, such as spectroscopy, where tabular methods outperform time series methods. Simple tabular models work better on some benchmark datasets than the selected time-series models. However, the time-series models are more accurate on some datasets and outperform tabular models with significant differences in accuracy. Tabular models are extremely fast and quickly scale to the number of time series. To further deepen our analysis, we extend the same experiment to the univariate time-series datasets and find that tabular models can outperform time-series models in accuracy. This implies that some problems can be solved using simple tabular models instead of time-series models, which are slower than tabular models. We discuss this for different problem domains and provide recommendations for researchers and practitioners.

In this thesis, we study the scalability of multivariate time series classifiers. We propose solutions to address the scalability challenges of classifiers with respect to time series length and number of channels. We also compare the performance of state-of-the-art time series models with simple tabular models. Our results show that our proposed

solutions can significantly improve the scalability of machine learning methods for analysing temporal data. Our work will make it possible to apply machine learning to more complex temporal data problems. In addition, we open-source all the code and data employed in this thesis.

Statement of Authorship

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

Signature _____

ACKNOWLEDGEMENTS

This thesis represents the journey of the past four years, marked by many ups and downs. While the text is primarily my work, it has benefited from the contributions of many others who have supported and guided me.

First and foremost, I express my deepest gratitude to Prof. Georgiana Ifrim for her unwavering support throughout this thesis. Georgiana, thank you for allowing me to work under your guidance and consistently pushing me to excel. You taught me the art of tackling complex problems and transformed me into a better researcher. Your encouragement and mentorship have been instrumental in my journey, especially during challenging times.

I also sincerely appreciate Dr. Thach Le Nguyen, who has been involved in this project since its inception and played a pivotal role in providing invaluable feedback and brainstorming ideas. I will always cherish your availability, especially during those late-night paper submission sessions.

I would also like to thank my research studies panel members, Prof. Mark Keane and Prof. Brian MacNamee. You both played a significant role in shaping the research questions and provided insightful feedback. Your unwavering support and critical guidance were invaluable to my success.

To my colleagues and friends, Ashish, Qinqin, Uche, Eoin, Ryan, Eduardo, Ayush, and everyone who has been a part of my PhD journey, I express my heartfelt gratitude for your unwavering support and camaraderie. I cherish the countless moments of laughter and shared experiences during our badminton and football matches. Your diverse perspectives, unwavering encouragement, and lessons learned from individuals of varying backgrounds have enriched my personal and intellectual growth.

I also sincerely thank Dr. Donagh Berry, Dr. Donnacha O'Driscoll, Cheryl Nolan, Rosemary Deevy, Olivia McCrum, and the entire Vistamilk, Insight, and Computer Science operations team for their invaluable logistical support. Your efforts in creating a conducive and supportive environment have made my stay at UCD an enriching and enjoyable experience.

Finally, I express my profound gratitude to my parents, who have always supported and inspired me. Their unwavering belief in my abilities has been the driving force behind my academic pursuits. I am also deeply indebted to my younger brother, whose selfless help and encouragement have been instrumental in my PhD journey.

This work was made possible through the generous support of Science Foundation Ireland (SFI) through the VistaMilk SFI Research Centre (SFI/16/RC/3835) and the Insight Centre for Data Analytics (12/RC/2289 P2).

LIST OF PUBLICATIONS

Primary publications emerging from the thesis:

- **Bhaskar Dhariyal**, Thach Le Nguyen, Severin Gsponer, and Georgiana Ifrim. "An examination of the state-of-the-art for multivariate time series classification." In 2020 International conference on data mining workshops (ICDMW), pp. 243-250. IEEE, 2020.
  https://doi.org/10.1109/ICDMW51313.2020.00042

- **Bhaskar Dhariyal**, Thach Le Nguyen, and Georgiana Ifrim. "Fast channel selection for scalable multivariate time series classification." In Advanced Analytics and Learning on Temporal Data: 6th ECML PKDD Workshop, AALTD 2021, Bilbao, Spain, September 13, 2021, Revised Selected Papers 6, pp. 36-54. Springer International Publishing, 2021.
  https://doi.org/10.1007/978-3-030-91445-5_3

- **Bhaskar Dhariyal**, Thach Le Nguyen, and Georgiana Ifrim. "Scalable classifier-agnostic channel selection for multivariate time series classification." Data Mining and Knowledge Discovery 37, no. 2 (2023): 1010-1054.
  https://doi.org/10.1007/s10618-022-00909-1

- **Bhaskar Dhariyal**, Thach Le Nguyen, and Georgiana Ifrim. "Back to Basics: A Sanity Check on Modern Time Series Classification Algorithms." In Advanced Analytics and Learning on Temporal Data. AALTD 2023. Lecture Notes in Computer Science, vol 14343. Springer, Cham.
  https://doi.org/10.1007/978-3-031-49896-1_14

Other contributions that were published during the course of this thesis:

- Maria Frizzarin, Antonio Bevilacqua, **Bhaskar Dhariyal**, Katarina Domijan, Federico Ferraccioli, Elena Hayes, Georgiana Ifrim et al. "Mid infrared spectroscopy and milk quality traits: A data analysis competition at the "international workshop on spectroscopy and chemometrics 2021"." Chemometrics and Intelligent Laboratory Systems 219 (2021): 104442.
  https://doi.org/10.1016/j.chemolab.2021.104442

- Maria Frizzarin, Giulio Visentin, Alessandro Ferragina, Elena Hayes, Antonio Bevilacqua, **Bhaskar Dhariyal**, Katarina Domijan et al. "Classification of cow diet based on milk Mid Infrared Spectra: A data analysis competition at the "International Workshop on Spectroscopy and Chemometrics 2022".Chemometrics and Intelligent Laboratory Systems 234 (2023): 104755. https://doi.org/10.1016/j.chemolab.2023.104755

CHAPTER

ONE

INTRODUCTION

1.1 Time Series Classification

Time series are data that are recorded as ordered sequences of numeric values. They are encountered in many applications, such as weather forecasting [4], financial market analysis [5] , and industrial process monitoring [6]. The proliferation of IoT and sensor technology has rapidly fueled the collection of such sequential data. The COVID-19 pandemic has also enhanced the use of wearable devices [7], which generate time series data about the wearer's health and activities. A Mordor Intelligence report [8] predicts that the wearable technology market size is expected to grow from USD 186.48 billion in 2023 to USD 419.44 billion by 2028. Figure 1.1 shows the popularity trend of different database types for the last three years. The time series databases are the clear front-runner among other databases. This is due to the increasing demand for time series data storage and analysis capabilities [9].

Time series applications vary across domains, e.g., sports science [2], agriculture [10], or health [11]. For example, nowadays, almost all fitness apps use phone sensors to measure acceleration(A), orientation(O), or angular velocity(ω) to identify an individual's motion, such as running, jogging or jumping. The sensors recording $[A, O, \omega]$ are known as channels in the time series literature, and the data recorded from multiple channels for the same period is known as Multivariate Time Series (MTS). The task of assigning a discrete label (e.g. running) to a single channel is called Univariate Time Series Classification (UTSC) while assigning a discrete label to an MTS is known as Multivariate Time Series Classification (MTSC).

[1] https://db-engines.com/en/ranking_categories

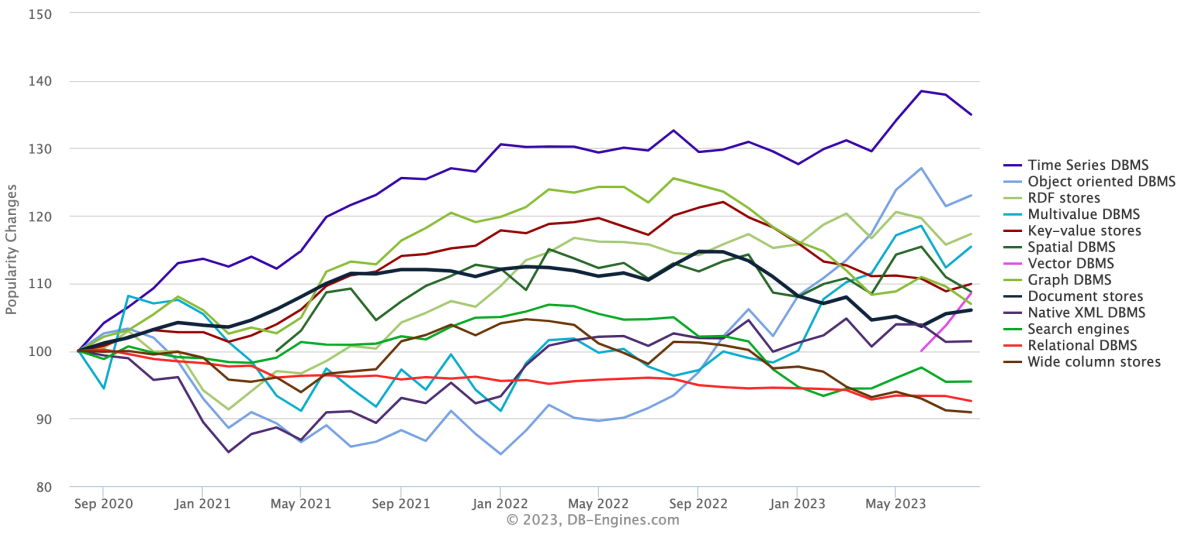


Figure 1.1: Growth of time series data over the last 36 months from DB-Engine[1].

1.2 Research Context

Research in UTSC has made significant progress [12]; however, a similar level of progress is yet to be achieved in MTSC. In the context of UTSC literature, MTSC is an extension of the conventional UTSC problem, leading to their adaptation to MTSC. Nevertheless, these adaptations often overlook critical factors such as inter-channel dependencies and computational complexities, such as time and space requirements. These factors are crucial for scalable MTSC solutions. Unfortunately, this oversight results in scalability issues. Scalability here refers to *the ability of an MTSC algorithm to handle large datasets and complex relationships between channels.*

The authors of study [13] compiled a comprehensive collection of existing multivariate time series classification (MTSC) datasets. They conducted benchmarking experiments using the 1NN-DTW [14] classifier to assess the algorithm's performance on these datasets. Similarly, another study [15] evaluated the WEASEL-MUSE classifier on the same benchmark datasets. Both studies extend their respective univariate classification techniques to MTSC. Both the 1NN-DTW and WEASEL-MUSE classifiers performed well on the MTSC datasets. However, those studies point out scalability issues despite their good classification performance. While these classifiers accurately categorise MTS data, they can be inefficient for large datasets or real-time applications. This scalability issue suggests that these methods while promising in terms of accuracy, need to be improved in their adaptability to scenarios where computational efficiency and quick processing are paramount. This prompts further exploration and develop-

ment of MTSC techniques that preserve accuracy and exhibit the scalability required for broader practical applications.

In this thesis, we comprehensively address the scalability challenges in multivariate time series classification (MTSC). We extensively evaluate the current state-of-the-art MTSC classifiers and propose novel methodologies to classify multivariate time series (MTS) data. We also propose new methodologies to improve the scalability of existing techniques. This work thoroughly examines various aspects of MTS data, such as the length and number of channels.

The following section discusses the questions we address in the various chapters. We discuss their importance and the potential impact they could have on scientific literature and the real world.

1.3 Research Questions & Contributions

- **What are the main scalability challenges faced by state-of-the-art MTSC algorithms?**

  To understand the scalability challenges of the current state-of-the-art (SOTA) algorithms for MTSC, we need to identify the root causes of these issues.

  First, we empirically evaluated the MTSC algorithms to identify the factors that limit their scalability, both in general and for individual classifiers. We found that improper handling of the number of channels and length of time series are significant contributors preventing them from scaling.

  Second, we proposed a new classifier that is more accurate and faster than the current baseline 1NN-DTW. Our classifier uses simple and efficient methods to extract features independently from each channel. We concatenate the features in a single feature space and train a linear classifier. Our results show that our classifier is as accurate as recent, more complex deep learning methods. This raises the question of whether there is any need to learn expensive and complex features or models that can capture complex dependencies across time series channels. Our experiments suggest that a linear combination of features extracted from each channel independently works well in terms of accuracy and is more efficient to train.

  The contributions are detailed in Chapter 3 and published in [16].

- **What is the impact of channel selection on the scalability of MTSC algorithms?**

By working on the previous question, we identified the length of time series and the number of channels as the two main issues that impact the scalability of classifiers. We proposed a method to address the scalability challenge posed by an increase in the length of time series. In this question, we address the scalability issue due to the number of channels.

The number of channels in a multivariate time series (MTS) dataset can contribute to longer computation times for MTSC algorithms. To address this, we proposed three greedy channel selection strategies for MTSC. We conducted extensive experiments on the UCR MTSC benchmark and showed that our strategies could achieve a 70% reduction in computation time while preserving classifier accuracy. We also showed that our strategies could achieve significant data storage savings, with up to 70% of the original data being discarded. We presented a case study of our methods on a real-world, 25-channel MTS dataset recorded for the Military Press strength and conditioning exercise.

The contributions in answering this research question are explained in Chapter 4 and published in [17].

- **How does noise in the data affect channel selection methods?**

  The greedy channel selection strategies discussed above can drastically reduce the computation time and memory required to store MTS datasets. However, these techniques are susceptible to noise, which can make them less effective.

  We propose new channel selection techniques that are accurate, scalable, and robust to noise. We conduct extensive experiments on the UEA MTSC benchmark and show that our techniques can achieve a 75% reduction in computation time and a 60% data reduction while preserving accuracy. We also demonstrate a significant improvement in the accuracy (more than 20%) of the state-of-the-art classifier ROCKET on synthetic datasets with varying difficulty levels in identifying important channels. In a real-world case study on a human motion MTSC dataset with 50 channels, our techniques achieve a significant accuracy improvement for ROCKET (more than 5%) when classifying the Military Press strength and conditioning exercise.

  The contributions in answering this research question are discussed in Chapter 5 and published in [18].

- **How do recent MTSC classifiers scale, and how do they perform against scalable tabular methods?**

  As mentioned earlier, the literature on time series analysis is constantly evolving as new data becomes available. In our previous studies, we empirically evalu-

ated state-of-the-art classifiers, proposed simple classifiers, and explored channel selection techniques. Since then, there have been significant developments in the time series classification literature. Notably, the emergence of models such as MiniRocket [19] and MultiRocket [20] has demonstrated remarkable performance that surpasses other state-of-the-art counterparts.

To address this question, we empirically compare these Rocket family classifiers to simple tabular models. This comparative exploration not only sheds light on their relative strengths and weaknesses but also leads us to unveil intriguing insights. We further expand the scope of our study by including univariate time series data as well. This "sanity check" helps us validate our findings' robustness across different data dimensions. Our comprehensive approach enables us to significantly contribute to the evolving discourse within the time series classification literature while providing a deeper understanding of the relative capabilities of various cutting-edge algorithms.

The findings in answering the research question are discussed in Chapter 6 and published in [21].

1.4 Thesis Outline

The remainder of the thesis is structured as follows.

- **Chapter 2** provides the necessary background and relevant literature on MTSC.
- **Chapter 3** empirically evaluates multivariate time series classification algorithms and proposes a simple and scalable classifier.
- **Chapter 4** proposes a novel method to perform channel selection and evaluates it on the main MTSC benchmark and a real-world dataset.
- **Chapter 5** extends on Chapter 4, proposes newer approaches to perform channel selection, and performs robustness analysis for the proposed channel selection approach on synthetic data and real-world military press dataset.
- **Chapter 6** performs the empirical evaluation of the state-of-the-art ROCKET family of classifiers on MTSC and UTSC datasets.
- **Chapter 7** concludes the thesis and provides future research directions.

In addition to contributing to the chapters above, we have released all of our open-source code in individual Github [2] repositories and popular libraries such as Sktime[3] and Aeon toolkit[4].

[2] https://github.com/mlgig

[3] https://github.com/sktime/sktime/blob/main/examples/channel_selection.ipynb

[4] https://www.aeon-toolkit.org/en/latest/examples/transformations/channel_selection.html

CHAPTER

TWO

RELATED WORK & BACKGROUND

2.1 Technical Background

In this section, we provide the technical definitions and notation required for our work.

2.1.1 Time Series

A univariate time series $X = (x_1, x_2, x_3, ...x_n)$ is an ordered sequence of n real values $x_i \in \mathbb{R}$ collected at regularly spaced intervals over a period of time. This means that the difference between any two consecutive values, also known as the sampling rate, is always constant. Several works deal with irregularly spaced time series; however, those are outside the scope of the current thesis. In this thesis, we focus on developing methods for time series with constant sampling rates.

In many applications, time series data is acquired from multiple sources simultaneously. When this occurs, the time series is represented as $X = (x_1, x_2, ..., x_m)$, where $m \in \mathbb{N}$ is the number of channels in time series such that $x_i = (x_{i1}, ...x_{in})$. If m=1, the time series is termed univariate; otherwise, it is categorized as multivariate. In this thesis, we focus on multivariate time series classification. For instance, if the time series represents human motion from a person's phone, then X is a multivariate time series describing motion in a particular direction. If the motion is recorded for 60 seconds, one value for each second, the length of the time series (n) is 60. And if the phone captures motion along the x, y and z-axis, it is said to have three channels. For example, Fig 2.1 shows the multivariate and univariate representation of Counter Movement Jump data [1].

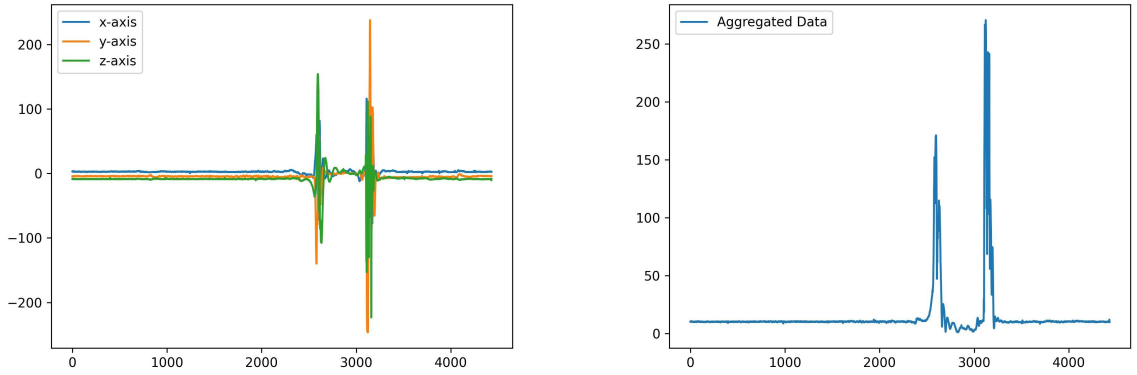


Figure 2.1: Illustration of Counter Movement Jump Data from [1]
(a) Shows the original accelerometer data. (b) Shows the summarized UTS values derived from the MTS measurements by computing the square root of the combined x, y, and z-axis measurements.

2.1.2 Multivariate Time Series Classification

The task of assigning a label to a time series is known as time series classification. When this task is applied to multivariate time series, it is called multivariate time series classification. A time series classification task involves learning a function $f : X \rightarrow Y$ where X represents a multivariate time series and Y is the corresponding label of time series such that $Y \in Z$.

This thesis addresses the challenges of multivariate time series classification, as outlined in the previous chapter. To effectively identify and propose solutions to these challenges, we utilize various methods from the existing literature. This chapter delves into a detailed description of those methods, particularly the classification algorithms and datasets employed in our research.

2.2 Multivariate Time Series Datasets

Data is fundamental for any machine learning research, and we also utilize standard benchmark datasets to assess and analyze algorithms. Throughout this thesis, we have implemented our proposed algorithms using the UCR/UEA [13] benchmark datasets. These datasets cover a wide range of domains, including human activity recognition (HAR), audio spectrum, motion, ECG, EEG/MEG, and other areas. Notable features include the longest time series, which measures 17894 values; the highest number of channels in the datasets is 1345; and the largest dataset, Insect Wingbeat, which comprises 30,000 samples in the train set and 20,000 in the test set. The data dictionary

for all datasets is provided in Table 2.1, where we highlight specific datasets and their unique characteristics.

2.3 Multivariate Time Series Classifiers

During the course of the study, we look to develop classifiers that are scalable and study the scalability of the current state-of-the-art classifiers.

2.3.1 1NN-DTW

1NN-DTW is a 1-Nearest Neighbour classifier with Dynamic Time Warping (DTW) distance and one of the most popular methods in MTSC. In [22], the authors proposed two versions of DTW for MTS data, DTW_I and DTW_D, to study the impact of DTW on multiple channels. The main difference between the two versions is how they compute the distance between two multivariate time series. The DTW_I assumes each channel of MTS as an independent univariate time-series and consequently sums up the distances for each channel pair. It calculates the optimal path P based on the point-wise distance between the time series. The DTW_D assumes that the correct warping is the same across all channels; it computes the distance between two-time series by first summing up the distance across each channel. Unlike for DTW_I, in DTW_D the optimal path P is based on the Euclidean distance between two vectors representing all channels. Although both versions of DTW have high accuracy and are considered a strong baseline for any MTSC task, they are computationally expensive[1] and have been outperformed in accuracy by more recent methods [23]. The number of channels and length of time-series/warping hyperparameter heavily impact both DTW_I and DTW_D; thus, optimising the number of channels can drastically help improve their scalability.

2.3.2 MrSEQL-SAX

MrSEQL-SAX [24], an extension of the SEQL [25] method originally designed for DNA sequence analysis, effectively tackles time series classification tasks. This linear classifier harnesses symbolic features extracted from time series data [26], transforming it into multiple symbolic representations across distinct domains (e.g., SFA [27] in the frequency domain and SAX [28] in the time domain) and various resolutions using

[1]We have evaluated here the sktime implementation of DTW.

Domains	Problems	Acronym	Train Size	Test Size	#Channels	TS Length	#Classes
Motion	**ArticularyWordRecognition**	AWS	275	300	9	144	**25**
ECG	AtrialFibrillation	AF	15	15	2	640	3
HAR	BasicMotions	BM	40	40	6	100	4
Motion	**CharacterTrajectories**	CT	1422	1436	3	182	**20**
HAR	Cricket	CKT	108	72	6	**1197**	**12**
Audio Spectra	**DuckDuckGeese**	DDG	50	50	**1345**	270	5
HAR	**EigenWorms**	EW	128	131	6	**17984**	5
Motion	Epilepsy	EP	137	138	3	206	4
Other	ERing	ER	30	270	4	65	6
HAR	**EthanolConcentration**	EC	261	263	3	**1751**	4
EEG/MEG	**FaceDetection**	FD	**5890**	**3524**	**144**	62	2
EEG/MEG	FingerMovements	FM	316	100	28	50	2
EEG/MEG	HandMovementDirection	HMD	160	74	10	400	4
HAR	**Handwriting**	HW	150	850	3	152	**26**
Audio Spectra	Heartbeat	HB	204	205	61	405	2
Audio Spectra	**InsectWingbeat**	IWB	**30000**	**20000**	**200**	30	**10**
Audio Spectra	JapaneseVowels	JV	270	370	12	29	9
Other	**Libras**	LIB	180	180	2	45	**15**
HAR	**LSST**	LSST	2459	2466	6	36	**14**
EEG/MEG	**MotorImagery**	MI	278	100	64	**3000**	2
HAR	NATOPS	NTP	180	180	24	51	6
Motion	**PEMS-SF**	PSF	267	173	**963**	144	7
Other	**PenDigits**	PD	**7494**	**3498**	2	8	**10**
Audio Spectra	Phoneme	PHM	3315	3353	11	217	39
HAR	RacketSports	RS	151	152	6	30	4
EEG/MEG	SelfRegulationSCP1	SR1	268	293	6	896	2
EEG/MEG	**SelfRegulationSCP2**	SR2	200	180	7	**1152**	2
Audio Spectra	SpokenArabicDigits	SAD	**6599**	**2199**	13	93	**10**
ECG	**StandWalkJump**	SWJ	12	15	4	**2500**	3
HAR	UWaveGestureLibrary	UWL	120	320	3	315	8

Table 2.1: Data Dictionary for 30 UCR Multivariate Time Series Classification datasets. The bold values represent unique datasets that stand out due to their distinctive characteristics, such as many channels, longer time series, a large number of data points, or a substantial number of data classes.

different window sizes. The classifier extracts discriminative subsequences from the symbolic representations, and these subsequences are later combined to form a single feature vector used to train a classification model. The method was initially developed for univariate time series classification but later adapted to MTSC; the adapted version views each channel as an independent time series. The symbolic transformations are computationally expensive and are bottlenecks for classifiers like MrSEQL-SAX and WEASEL-MUSE, discussed below. Furthermore, the transformation over multiple windows iterating over full-time series incurs a high cost for the scalability of the classifier. Therefore, reducing the number of channels can significantly improve the classifier's scalability. This study focuses on MrSEQL-SAX (MrSEQL using only SAX features) as it is more efficient than the variant combining SAX and SFA transformations.

2.3.3 WEASEL-MUSE

WEASEL-MUSE [29] is an extension of the WEASEL algorithm [30] to MTSC data. The classifier builds a bag-of-patterns (BOP) model using the SFA transform for every channel. It captures the patterns by rolling multiple windows of varying sizes on raw and derivative time series and transforming those segments into unigram and bigram words. The classifier links these words to their respective channel and creates a histogram for each channel separately. To deal with the large number of features from every channel, it uses the Chi-square feature selection method to remove the irrelevant features. The selected features are concatenated into a single feature vector, which is the input to a logistic regression algorithm. Like MrSEQL, the WEASEL-MUSE classifier also iterates over the entire time series for every channel and performs the SFA transform for every window. This iteration and transformation increases the overall computation cost. Also, storing many unigrams and bigrams in memory is quite expensive. Therefore, we expect that channel selection would positively impact this classifier.

2.3.4 ROCKET Family

There are three classifiers in the ROCKET family, namely ROCKET, MiniROCKET and MultiROCKET.

2.3.4.1 ROCKET

ROCKET [31] is one of the most accurate time-series classifiers, initially proposed for univariate time-series classification and later extended to MTSC. ROCKET relies on convolutional kernels to extract time-series features. The kernels in ROCKET are generated randomly from a Gaussian distribution; however, unlike in Convolutional Neural Networks [32] they are static throughout the classification process. The kernel properties like length, dilation, stride, bias and zero-padding are sampled randomly. The MTSC version of ROCKET also performs channel selection randomly, selecting a maximum of 12 channels for any MTSC dataset from the UCR benchmark. Therefore, the runtime is not significantly affected by the number of channels, especially for datasets with more than 12 channels.

The kernels are a linear transformation of the input time series. A linear classifier, ridge regression, trains on the feature vector formed by global max pooling and the proportion of positive values (PPV) features extracted from the convolution time series from every channel. ROCKET has become very popular due to its high accuracy and speed, yet the impact of channel selection on this classifier in the MTSC task has not been examined extensively.

2.3.4.2 MiniROCKET

The authors of MiniROCKET [19] addressed the randomness of ROCKET's kernels, which can negatively impact classification robustness. They achieved this by reducing the number of kernels from 10000 in ROCKET to a significantly smaller set of 84 deterministic kernels. Additionally, they removed the pooling operation from the classifier, further enhancing performance. The authors' efforts resulted in improved accuracy and reduced computational time for their classifiers.

2.3.4.3 MultiROCKET

MultiROCKET [20] is a fast and effective time series classification algorithm that achieves state-of-the-art accuracy while significantly reducing computation time compared to other TSC methods. It builds upon MiniROCKET by introducing multiple pooling operators and transformations to extract more diverse and informative features from time series data. The authors use multiple pooling operators, first-order differences, and various transformations to diversify the generated features.

2.3.5 Deep Learning

2.3.5.1 Time Series Attentional Prototype Network (TapNet)

Tapnet [33] is one of the models that deals exclusively with multivariate time series data. The model proposes an attention prototype network that takes inspiration from traditional and deep learning frameworks. The model has three components, namely, Random Dimension Permutation, Multivariate Time Series Encoding and Attentional Prototype Learning. The authors propose Random Dimension Permutation (RDP) to reorganize the channels into groups. From these groups, a low-dimension embedding for the sequences is learned using an LSTM and a one-dimensional CNN. Using this embedding, a class prototype is learned, where a class prototype is a weighted combination of training samples belonging to the same class, and the weights are learned using an attention layer. The TapNet classifier requires hyper-parameters to be tuned separately for each dataset in the benchmark, which tends to overfit the test set.

2.3.5.2 Multivariate Long Short Term Memory- Fully Convolutional Network (MLSTM-FCN)

Multivariate LSTM-FCN [34] builds upon LSTM-FCN and Attention LSTM-FCN models. It incorporates a convolutional block followed by an LSTM block for sequence processing. The convolutional block is similar to the base models, using a convolution layer, batch normalization, and the ReLU activation function. The key difference lies in the addition of a squeeze-and-excite block. This block dynamically adjusts the importance of each channel within the data. The squeeze operation analyzes the relationships between features, while the excite operation emphasizes informative features and suppresses less important ones. This helps the model focus on the most relevant aspects of the multivariate time series data for improved classification.

2.3.5.3 Transformers

The remarkable success of the transformer architecture [35] in computer vision and natural language processing has spurred the exploration of its applications in time series analysis. However, transformers have primarily found success in forecasting tasks, while their performance in classification tasks remains limited. This discrepancy could be attributed to the contrasting training requirements of forecasting and classification. In forecasting, self-supervised learning allows models to access vast amounts of data, enabling them to learn temporal patterns effectively. Conversely, time series classific-

ation necessitates labelled data, which becomes increasingly challenging to obtain in large quantities. Due to their deep learning nature, transformers demand substantial amounts of data for optimal performance. This scarcity of labelled time series data poses a significant hurdle to their effective application in classification tasks. Nevertheless, ConvTran [36] is the latest transformer model based on multivariate time series classification. The authors propose two position encodings, namely, time absolute position encoding (tAPE) and efficient relative position encoding (eRPE) for MTSC. The encoded data is then passed to the transformer model, where the downstream task is classification. Apart from ConvTran, Cross Attention Stabilized Fully Convolutional Neural Network (CA-SFCN) [37], Transformer-based Framework (TST) [38], BErt-inspired Neural Data Representations (BENDR) [39] are attention based multivariate time series models. The bottleneck of transformer models in our study is their computational requirement. Each of the transformer models requires heavy computation infrastructure and access to GPUs.

Apart from the classifiers discussed in Chapter 3, the authors in [23] also conducted a detailed study on multivariate time series classification methods. The authors use 1NN-DTW [40] and several other classifiers namely Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [41], Canonical Interval Forest (CIF) [42] and Generalized random shapelet forest (gRFS) [43] and Weasel Muse, MrSEQL and two deep learning methods namely, InceptionTime [44], TapNet. The same authors also proposed HIVE-COTEv2 [45], which is extended and updated version of HIVE-COTE classifier. Table 2.2 illustrates the characteristics of some of the prominent classifiers. While most of the column meanings are self-sufficient, the Adapted column implies the classifiers were adapted from univariate to multivariate forms.

Classifiers	Univariate	Multivariate	Adapted	Ensemble	GPU
1NN-DTW	✓	✓	✓		
MrSEQL-SAX	✓	✓	✓		
WEASEL-MUSE	✓	✓	✓		
TapNet		✓			✓
MLSTM-FCN		✓			✓
ROCKET	✓	✓	✓		
MiniROCKET	✓	✓	✓		
MultiROCKET	✓	✓	✓		
InceptionTime	✓	✓		✓	✓
HIVE-COTEv2	✓	✓	✓	✓	
ConvTran		✓			✓

Table 2.2: Summary table of various prominent classifiers

2.4 Channel Selection

There's a significant amount of research in machine learning aimed at reducing the number of features used in tabular data. This is typically achieved through two main approaches: dimension reduction and feature selection.

Dimension reduction: In this approach, the original set of features is transformed into a new, lower-dimensional feature space before being fed into a classification algorithm. **Feature selection**: Here, the original feature vector remains intact, but only a relevant subset of features is chosen for further analysis. This approach aims to identify the most informative features that contribute most to the prediction task. There are three primary methods for feature selection: filter, wrapper, and embedded [46].

Filter Methods works independently of any model, analyzing features themselves (like correlation, covariance) to pick relevant features. These methods are fast and easy, but are prone to miss feature interactions. Examples of filter methods are variance threshold and Pearson's correlation. *Wrapper Methods* wraps around a chosen model, testing different feature subsets and selecting the one that performs best. These methods are more accurate potentially, but are computationally expensive and prone to overfitting. The examples of wrapper methods are forward selection and backward elimination. *Embedded Methods* feature selection is built into the model training process itself. It offers a balance between filter methods' speed and wrapper methods' model-specific selection. The examples embedded methods include LASSO and Ridge regression.

Similar to tabular methods in machine learning, channel selection for multivariate time series is a recurring topic in the MTSC literature. However, the focus of most work has been on accuracy rather than scalability.

The most recent work on channel selection [47, 48] aims to identify the best subset of channels. The proposed method calculates a merit score based on correlation patterns of the outputs from the classifiers. The algorithm iterates through every possible subset to calculate the merit score, followed by a wrapper search on the subset with the top 5% merit score. 1NN-DTW is employed as the wrapper to perform the classification. The computational bottleneck of this work is using DTW over every possible subset to identify useful channels.

Another notable study is CleVer [49], where the author proposed three unsupervised feature subset selection techniques employing Common Principal Component Analysis (CPCA) [50] to measure the importance of each sensor. The authors build a correlation coefficient matrix among different channels for each MTS. The principal

components of each coefficient matrix are calculated, all the principal components are aggregated together, and descriptive component principal components are calculated. The l_2-norm of the resulting vector generates the rank of each channel.

The authors in [51] proposed a framework for channel selection using a voting-based method. The two criteria used were distance-based classification and confidence-based classification. These methods were proposed for streaming data, which is outside the scope of the current study.

A recent study [37] presents an algorithm for channel ranking and channel selection. The channel ranking algorithm assigns a relevance score for each channel. The relevance scores are constructed on a similarity graph among the channels. The authors find the largest eigenvector of the normalized adjacency matrix of the similarity graph, which reflects its cluster structure. Apart from channel ranking, the authors also propose channel subset selection. From the adjacency matrix above, the algorithm finds the linear combination of matrices that approximates the labels' similarity matrix and uses the minimum number of redundant channels. Although the proposed channel ranking and selection approach performs well regarding accuracy, it is slow. The authors mention that they face computation issues while building the graph representation for each feature.

In a recent study, [52] evaluated and compared the channel selection methods proposed in [17] and [47] for the HIVE-COTEv2 [45] architecture. The authors proposed novel channel selection techniques based on the merit score metric with modifications to the existing methods. Specifically, they replaced the Dynamic Time Warping (DTW) algorithm with ROCKET. They trimmed the search space from 2^m to a smaller set using a forward selection method until the merit score plateaued. Additionally, they introduced CLUSTER, a modified version of the ECP and ECS methods discussed in Chapter 4, where the distance between each instance and the centroid was used instead of the centroid distance between two classes. The authors found that ECP and MSTS can reduce the number of channels in the dataset without significantly reducing the accuracy of HIVE-COTEv2. However, authors note that they would prefer ECP over MSTS, as it closely recreates HIVE-COTEv2 results.

While the aforementioned methods effectively select informative channels, they often come at the expense of increased computational time, sometimes leading to a substantial overhead compared to the default classifiers without channel selection. Notably, most of these methods lack publicly available code, hindering their direct evaluation and comparison. The one exception is [47], whose code was recently released. However, their primary focus is on improving classifier accuracy without considering scalability, which is a critical aspect of our research goal.

In order to compare our methods to a strong baseline, we adapt the feature selection methods from scikit-learn[2] to work with multivariate time series, by considering each channel as a single feature. We also investigate another baseline that uses the l_2-norm of each channel as a criterion for ranking and selecting channels in Chapter 4.

Criteria for method selection: We recognize that numerous classifiers and channel selection methods exist for multivariate time series classification in the literature. Practical considerations primarily guided our selection process. While the availability of open-source code from authors or reputable libraries was the primary factor, the publication date of the relevant papers also played a significant role.

To ensure a fair and unbiased comparison among various classifiers and channel selection methods, our experiments primarily focus on those with open-source code that can be executed on similar and available computing environments. This approach is crucial as deep learning architectures, which often rely on specialized hardware like GPUs, introduce an uneven playing field, potentially compromising the evaluation of different methods. By restricting our analysis to classifiers and channel selection methods with equal computational resources, we aim to provide a more objective assessment of their respective strengths and limitations.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html

CHAPTER

THREE

AN EXAMINATION OF THE STATE-OF-THE-ART FOR MULTIVARIATE TIME SERIES CLASSIFICATION

3.1 Introduction

The widespread adoption of IoT and sensor technology has made it much easier to collect time-series data. For example, a simple mobile phone can now simultaneously record data on acceleration (A), orientation (O), and gyroscopic movement (G) while a person is moving. This collection of multi-channel data, represented as a sequence of values [A, O, G], captured over a period of time, is known as multivariate time series (MTS) data.

When discrete labels are assigned to the MTS data, the task is known as multivariate time series classification (MTSC). While much research has been done on classifying univariate (single-channel) time series data (UTSC), the challenge of MTSC has received less attention. Generally, the UTSC methods are repurposed for solving MTSC problems. The study conducted by [13] evaluated the adaptation of the classic 1-nearest neighbour with Dynamic Time Warping (1NN-DTW) method, which was originally designed for UTSC, to the MTSC problem.

While converting multi-channel data to single-channel data is simple and effective, it can miss information only available when taking the interaction of multiple channels into account. Additionally, MTSC problems can be computationally expensive, as the overall length of the time series is increased by $c \times l$, where c and l stand for the number of channels and length of time series, respectively.

Moreover, while exploring the existing methods in the literature, we noted a significant challenge: the need for a unified platform for comparing different studies. Specifically, the published papers we surveyed did not consistently compare their methods on a standard set of benchmark datasets. Instead, most works introduced their methods using different datasets, making conducting a thorough and impartial evaluation difficult. This chapter aims to bridge this gap. We evaluate and contrast the various methods present in the literature by evaluating and comparing existing methods on the UEA MTSC benchmark [13].

We first identify some of the most prominent TSC methods and evaluate their classification performance. In particular, we examine the state-of-the-art algorithms, DTW [13], ROCKET [31], MrSEQL [24] as well as two recent deep learning approaches, Tapnet [33] and MLSTM-FCN [53]. Furthermore, we introduce new simple baseline methods that rely on statistical features extracted over the time series. Second, we discuss the advantages and drawbacks of the methods evaluated and give insights into which algorithm to use based on desired properties. For example, besides predictive accuracy, some of the compared methods only work on fixed-length time series, some have challenges with regard to memory consumption or runtime, and some of the methods are non-deterministic and are very complex, which makes it difficult to reproduce results or to provide a prediction explanation to an end user.

Our overarching goal is to analyse the current state-of-the-art in MTSC and then investigate methods to improve their accuracy with reduced computational complexity. Furthermore, we prefer conceptually simpler methods to understand and implement because, from our experience, this is desirable for real applications.

The main contributions of this chapter are as follows:

- We present an empirical study that compares current state-of-the-art MTSC methods on the UEA MTSC benchmark regarding both accuracy and computation time.
- We propose a simple baseline method for the MTSC task, which is faster and more accurate than the classic DTW baselines.
- We show that simple and efficient methods that extract features independently from each time series channel, concatenate them in a single feature space and train a linear classifier are as accurate as recent, more complex deep learning methods. This raises interesting questions about whether there is any need to learn expensive and complicated features or models that can capture complex dependencies across time series channels. From our experiments, a linear com-

bination of features extracted from each channel independently seems to work well regarding accuracy and is more efficient to train.

The rest of the chapter is structured as follows. In Section 3.2, we introduce our new baseline methods and explain their different variants. In Section 3.3, we present the UEA MTSC benchmark used for the experiments and discuss the empirical results. We conclude in Section 3.4.

3.2 Methods

The state-of-the-art MTS classifiers are usually complex and heavily engineered. Additionally, most of them suffer from scalability issues. In this part, we introduce some simpler techniques relying on basic statistics that can be directly extracted from the time series. We build our various methods with increasing complexity in order to understand the effect of data pre-processing steps on the accuracy and efficiency of these methods. Later we compare these approaches to the DTW baselines and to state-of-the-art MTSC methods.

3.2.1 Proposed Baselines

The fundamental principle of all the methods we propose here is to extract features from each channel of the time series independently. The resulting features are then concatenated to generate a feature vector for the time series, which later can be used with any classification algorithm that accepts feature vectors as input. The reason for this approach is that time series channels can be processed in parallel for extracting basic statistics. Collecting the features extracted from each channel into a single feature vector, allows us to learn dependencies between channels, without a heavy computational burden, as incurred by methods that try to directly model channel dependencies.

Classifier For simplicity and scalability, when training TSC models, we limit ourselves to linear models. These are known to be effective with large features spaces and can be trained efficiently. We employ the Ridge Regression Classification algorithm as implemented in the sklearn [54] library, as it is a linear classifier and has shown promising results in [31].

Our first approach computes four simple statistics over each channel of the time series at hand. Namely, we compute the mean, minimum, maximum and standard deviation for each channel independently. The resulting feature vector is $(4 * d)$-dimensional,

where d is the number of channels in each time series. We henceforth call this approach 4_stat. To capture more patterns without greatly increasing the complexity of the method, we add five additional properties that can be computed efficiently from the time series. These are the median, kurtosis, number of peaks, 25 and 75 percentile resulting in a total of 9 features per channel. Consequently, we term this method 9_stat. Finally, we use an even more extensive set of statistical features named Catch22; these features were found to be effective in the extensive empirical study [55]. We showcase the general pipeline of these simple approaches in Figure 3.1.

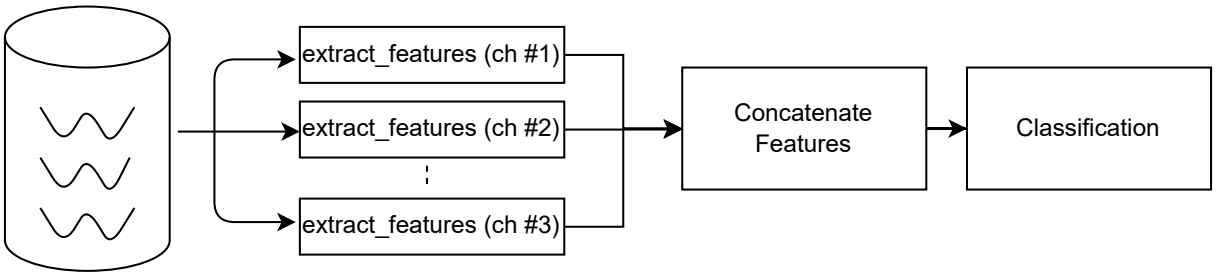


Figure 3.1: Feature extraction and classification in our proposed baselines.

Furthermore, we explore the effectiveness of the feature sets mentioned above while utilizing two techniques often employed in the time-series classification domain. First, to be able to capture details at a finer level, we use a rolling window approach. For

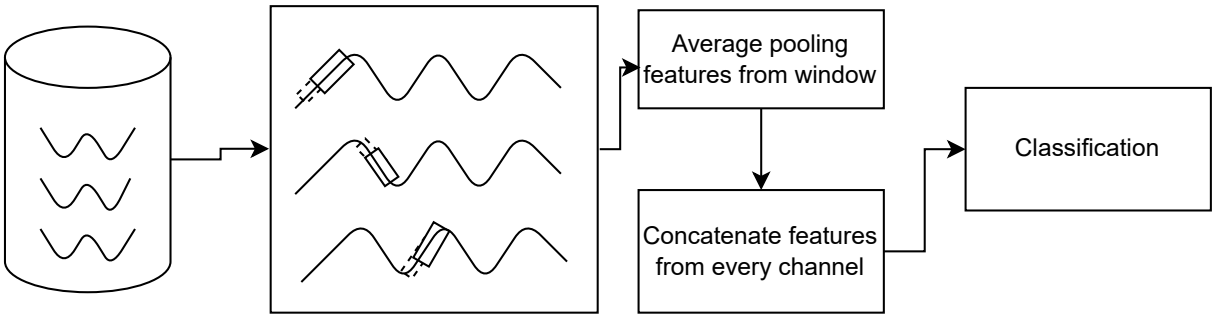


Figure 3.2: Feature extraction and classification using a rolling window.

each window, we compute the stats-features, followed by an average pooling step for each channel and each feature. The average pooling step aims to get rid of noise and anomalies in the data. One thing to note is that the average (mean) feature will be exactly the same as if it is computed on the non-windowed time-series. We showcase this approach in Figure 3.2. Some of the Catch22 features do not make sense in such a setting, due to short windows and dimension explosion, and hence we do not apply this technique while extracting Catch22 features. Secondly, we reduce the length of the time series using Piecewise Aggregate Approximation (PAA) [56]. PAA is a common dimensionality reduction technique used for time series and leads to shorter and smoother

Method	4 stats	9 stats	Catch22	window	PAA
4_stat	x				
9_stat		x			
Catch22			x		
4_stat_window	x			x	
9_stat_window		x		x	
4_stat_PAA	x				x
9_stat_PAA		x			x
Catch22_PAA			x		x
4_stat_MulPAA	x				x
9_stat_MulPAA		x			x

Table 3.1: Summary of proposed baselines. All methods extract features from each time series channel, concatenate all features in a single feature vector and train a linear classifier.

time series. In Figure 3.3 we show an example of extracting features from three different PAA transformed time series, where, for example, the number in PAA_30 means that we reduce the length of the time series to 30% of its initial length. This number is an input parameter to the PAA transform. Besides the simplicity and ease of under-

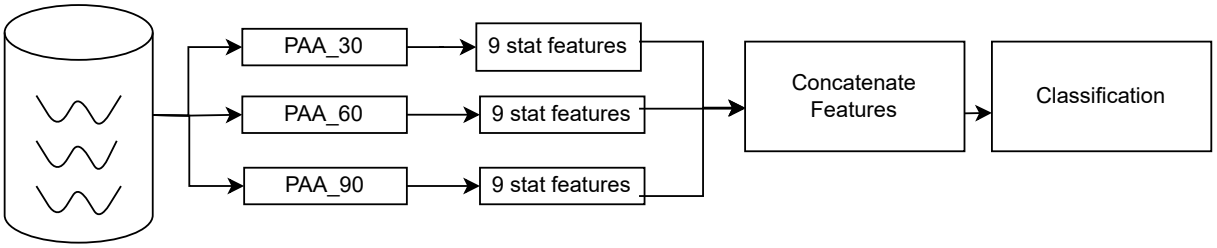


Figure 3.3: Feature extraction and classification using multiple PAA.

standing of the features, a major advantage is that all of the features for our proposed baselines can be computed efficiently and in parallel, since the time series channels are considered independently during the feature extraction step. Table 3.1 provides a schematic overview of all the introduced methods and their corresponding names. The code for our methods is available on Github[1].

3.3 Experiments

In this section we describe the MTSC benchmark used for our evaluation and present the methods compared and their results.

[1] https://github.com/mlgig/mtsc_benchmark

Data -set	9_stat MulPAA	DTW$_I$	DTW$_D$	WEASEL MUSE	MLSTM FCN	TapNet	Rocket MTS	MrSEQL SAX
AWR	0.99	0.98	0.99	0.99	0.79	0.97	0.99	0.99
AF	0.33	0.27	0.20	0.40	0.27	0.20	0.07	0.27
BM	0.98	1.00	0.98	1.00	1.00	1.00	1.00	0.95
*CT	0.96			0.99				0.96
CR	0.99	0.99	1.00	0.98	0.82	0.93	1.00	0.99
DDG	0.58	0.48	0.58		0.54	0.52	0.52	0.34
ER	0.91	0.92	0.92	0.97	0.87	0.88	0.99	0.88
EW	0.66	0.60	0.62		0.71		0.91	
EP	0.97	0.98	0.96	0.99	0.80	0.96	0.99	0.99
EC	0.28	0.30	0.32	0.47	0.25		0.42	0.56
FD	0.56	0.51	0.53	0.63	0.55		0.64	
FM	0.54	0.52	0.53	0.55	0.56	0.44	0.51	0.56
HMD	0.31	0.29	0.19	0.37	0.34	0.39	0.45	0.15
HW	0.18	0.51	0.61	0.52	0.50	0.32	0.58	0.47
HB	0.77	0.66	0.72	0.71	0.66	0.55	0.75	0.73
*IWB								0.10
*JV	0.97			0.98				0.45
LSST	0.49	0.58	0.55	0.64	0.75	0.55	0.64	0.59
LIB	0.68	0.89	0.87	0.89	0.64	0.82	0.91	0.87
MI	0.47	0.01	0.50		0.51		0.58	0.51
NTP	0.90	0.85	0.88	0.91	0.96	0.91	0.88	0.87
PSF	0.82	0.73	0.71		0.99	0.80	0.84	0.96
PD	0.82	0.94	0.98	0.97	0.68	0.97	0.98	0.92
PS	0.16	0.15	0.15		0.27	0.13	0.27	0.26
RS	0.87	0.84	0.80	0.93	0.82	0.85	0.91	0.87
SRS1	0.77	0.77	0.78	0.70	0.87		0.86	0.68
SRS2	0.44	0.53	0.54	0.53	0.52	0.49	0.53	0.51
*SAD	0.95							0.98
SWJ	0.33	0.33	0.20	0.27	0.33	0.33	0.53	0.33
UW	0.73	0.87	0.90	0.93	0.75		0.94	0.87
Wins/Ties	0	3	3	4	4	1	14	2
Completed	29	26	26	23	26	20	26	28

Table 3.2: Accuracy results of all compared methods on all the MTSC benchmark datasets where the method could complete training and prediction. Some of the classifiers are not able to handle variable length time-series; the dataset with variable length time series are marked *.

Data -set	9_stat MulPAA	DTW$_I$	DTW$_D$	WEASEL MUSE	MLSTM FCN	TapNet	Rocket MTS	MrSEQL
AWR	0.62	2.28	0.40	179.00	1.39	2.67	0.13	8.70
AF	0.01	0.05	0.40	0.25	0.81	1.86	0.01	0.28
BM	0.07	0.03	0.30	0.10	1.03	0.03	0.01	0.26
*CT	0.83			121.38				23.04
CR	0.58	7.35	2.51	4.51	1.77	1.69	0.34	16.44
DDG	17.17	44.51	6.48		40.55	7.01	0.77	219.45
ER	0.14	0.04	0.40	0.09	0.85	1.23	0.03	0.16
EW	8.40	11386.00	2769.30		135.19		14.65	
EP	0.15	0.29	0.17	2.71	0.67	7.74	0.13	0.82
EC	1.07	60.49	35.20	72.02	2.87		0.77	31.07
FD	89.80	2339.00	272.13	514.00	66.12		5.49	
FM	0.92	0.27	0.10	0.52	2.22	0.06	0.08	4.11
HMD	0.48	3.12	1.10	28.35	0.93	10.71	0.11	14.34
HW	4.98	72.46	8.42	4.42	0.68	47.85	0.63	42.97
HB	0.37	1.17	0.58	27.24	5.70	0.02	0.09	1.89
*IWB								150.92
*JV	0.55			0.57				0.19
LSS	1.90	9.13	3.10	24.10	1.40	139.69	0.30	21.19
LIB	0.07	0.40	0.40	0.54	6.68	7.10	0.03	0.24
MI	22.80	2295.00	418.00		15.28		3.20	992.61
NTP	0.68	0.23	0.80	8.14	2.20	5.39	0.07	2.85
PSF	37.03	156.33	21.33		11.93	16.76	1.78	600.94
PD	1.35	1.00	1.12	0.41	136.50	111.08	1.41	0.94
PS	8.44	1010.28	196.48		11.48	164.96	1.09	1030.10
RS	0.16	0.04	0.30	0.31	0.68	3.62	0.02	0.26
SRS1	1.23	39.50	15.10	36.72	1.88		0.49	20.55
SRS2	1.20	47.57	15.43	45.73	1.95	0.06	0.70	22.99
*SAD	7.16							119.29
SWJ	0.15	0.57	0.37	5.62	0.62	0.00	0.04	2.16
UW	0.30	1.32	1.04	9.59	0.68		0.14	1.87
Total time	208.62	17478.43	3770.96	1086.32	452.08	529.53	32.51	3330.64

Table 3.3: Total training time in minutes for all compared methods on all the MTSC benchmark datasets where the method could complete training and prediction. Some of the classifiers are not able to handle variable length time-series; the dataset with variable length time series are marked *.

3.3.1 MTSC Datasets

All approaches discussed in this chapter were evaluated on the UEA MTSC archive [13]. There are a total of 30 MTSC datasets which come from various domains, including human activity recognition, motion classification, ECG classification, EEG/MEG and audio spectra classification. The datasets have high variety in terms of the number of time series channels, sample size and length. The channels vary from 2 to 1,345, while the training sample sizes range from 12 to 30,000. In this chapter, we specifically focus on handling time-series length since the archive contains many time-series lengths ranging from 8 to 17894. Additionally, the archive also features some variable-length data. We use the data as is, although many of the compared methods have an intermediate step of normalising the data before training a model.

We use the critical difference (CD) diagram [57] to compare multiple MTSC methods. The CD diagram is a visual representation of the mean accuracy rank of multiple classifiers computed over multiple datasets. It relies on pairwise statistical significance testing of the classification accuracy of compared methods and connects with a thick horizontal line method that is not found to have a statistically significant difference. As recommended in [57, 58, 59], we first perform a Friedmann test, followed by a Wilcoxon signed-rank test with Holm correction. We use the R library `scmamp` [60] to compute and plot all the CD diagrams.

3.3.2 Results and Discussion

In this study, we consider DTW_I and DTW_D as our baseline models. We were able to reproduce the results for these baselines on 26 datasets that have time series of equal length. The DTW_I and DTW_D implementation[2] that we used in the experiments cannot handle variable-length time series. Some of the methods, such as 9_stat_MulPAA and MrSEQL-SAX, also work with variable-length time series. All comparative analysis in our experiments is done on these 26 datasets unless mentioned otherwise (e.g., some methods ran into memory issues due to the scale of the data). All the software implementations of the compared methods were run on a shared server. The server has 512GB memory with 72 cores and 32GB VRAM Nvidia GTX1080Ti. It operates on an Xeon E5-2695 processor with Ubuntu 18.04 LTS. We provide detailed accuracy and runtime results for all state-of-the-art classifiers in Table 3.2 and Table 3.3.

[2] https://github.com/uea-machine-learning/tsml

3.3.2.1 Comparing Variants of our Proposed Baselines

We first measure the quality of the models based on statistics-features for classifying multivariate time series. As described in Section 3.2, we start by extracting 4 stats, 9 stats or Catch22 features from each channel of the time series and compare the accuracy of the resulting models. From Figure 3.4, we see that the Catch22 feature set is the most effective in extracting the important information from the various channels. It is interesting to see that more stats features contribute to better accuracy.

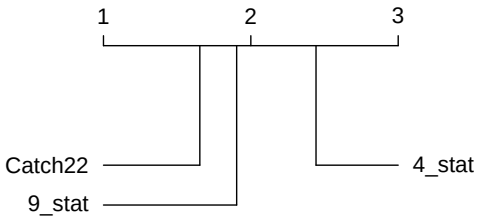


Figure 3.4: Critical difference diagram for methods based only on statistical features on 26 datasets.

In the next set of experiments, we investigate the effect of techniques used to smooth the input data (PAA) as well as to capture features at different resolutions (windowing). The results of all combinations can be found in Figure 3.5. We note that the best combination method is the one that uses 9 stats features with multiple PAA representations, although the difference to some of the other methods is not statistically significant.

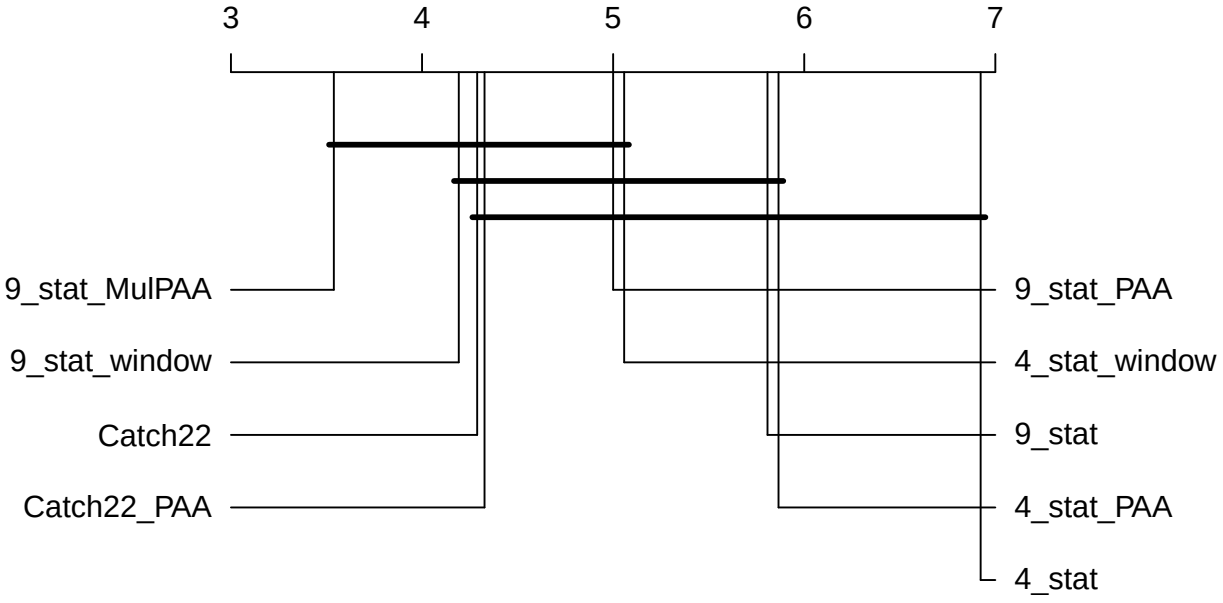


Figure 3.5: Comparison of all proposed stats-based baselines on 26 datasets.

Overall, the experiments show that both windowing and smoothing are effective as they both can improve the performance of their respective models. Furthermore, windowing appears to be more effective than PAA, as shown with the 4_stat and 9_stat models. Nonetheless, PAA is surprisingly ineffective when combined with Catch22 features. Finally, using multiple levels of smoothing ($\_MulPAA$ methods) had the strongest impact as it elevates the 9_stat model to the top of the average ranking. We do not combine Catch22 with multiple PAA due to efficiency issues since computing the Catch22 features takes too long to run multiple times, and this runtime increases with the number of times PAA is being applied.

With regard to comparison to the state-of-the-art, we start by comparing our best method, 9_stat_MulPAA, to the DTW_D and DTW_I baseline methods introduced with the MTSC benchmark [13]. After performing a Friedmann test, we found no statistically significant difference in accuracy between these three methods. Figure 3.6 gives a more detailed overview of the pairwise accuracy of the compared methods over the 26 datasets. The key difference, though comes from the computational efficiency of the stats-based baseline 9_stat_MulPAA versus the DTW methods. The running time of these methods is reported and discussed in Section 3.3.2.3.

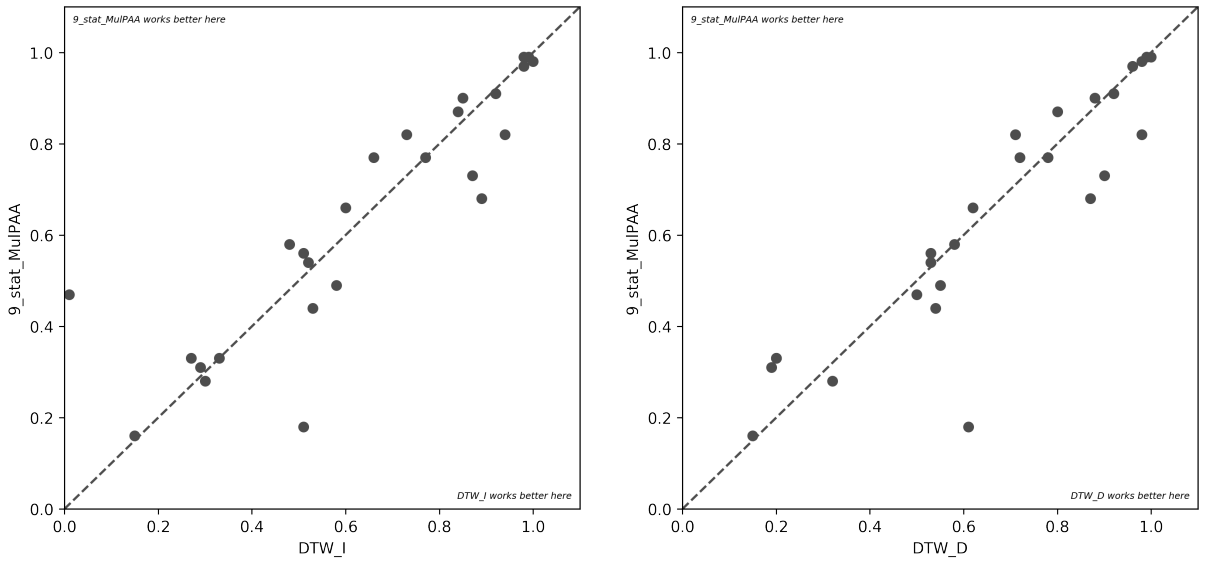


Figure 3.6: Accuracy comparison between (a) 9_stat_MulPAA vs DTW$_I$ and (b) 9_stats_MulPAA vs DTW$_D$

3.3.2.2 Comparison with State-of-the-art (SOTA)

In this section, we compare our proposed method with the SOTA algorithms using a critical difference diagram between all the classifiers. All the SOTA methods (Rocket-

	RocketMTS	WEASEL-MUSE	MrSEQL-SAX	DTW_D	DTW_I	9_stat_MulPAA	MLSTM FCN
Average Rank	**2.57**	2.97	4.22	4.30	4.52	4.67	4.72
Wins/Ties	**9**	3	5	3	1	0	5

Table 3.4: Average rank and wins/ties for 7 compared methods across 20 datasets where all methods could run and deliver results.

MTS [3], MLSTM-FCN [4], WEASEL-MUSE [5], MrSEQL-SAX [6] (MrSEQL with SAX transformation only), TapNet [7]) results are obtained by running the original code provided by the authors. Unfortunately, although we followed the instructions in the original papers, we could not reproduce results for some of the methods compared. In the case of MLSTM-FCN, we opted to set the hyperparameter values recommended by [61], which resulted in better results. Some classifiers cannot handle variable-length time series (e.g., DTW, ROCKET, MLSTM-FCN), while others ran into memory issues (e.g., MUSE ran out of memory) or did not produce any results for some datasets (e.g., TapNet, MLSTM-FCN). The diagram in Figure 3.7 is based on 20 datasets for which we were able to obtain results with all methods.

RocketMTS is the most accurate algorithm, followed by WEASEL-MUSE and MrSEQL-SAX. However, WEASEL-MUSE frequently runs out of memory. After a bit of a gap, Mr-SEQL comes third. Compared to RocketMTS, the latter two methods have the advantage that they can also work with variable-length time series; nonetheless, they take more time for longer time series. The top three methods are followed by DTW_D, DTW_I and 9_stat_MulPAA, which are described in the earlier section. Surprisingly, the last ranked method is the deep learning approach MLSTM-FCN. The other deep learning method we analysed, TapNet, is not included in this comparison since it only completed on 17 out of these 20 datasets. The results for Tapnet are given in our code repository.

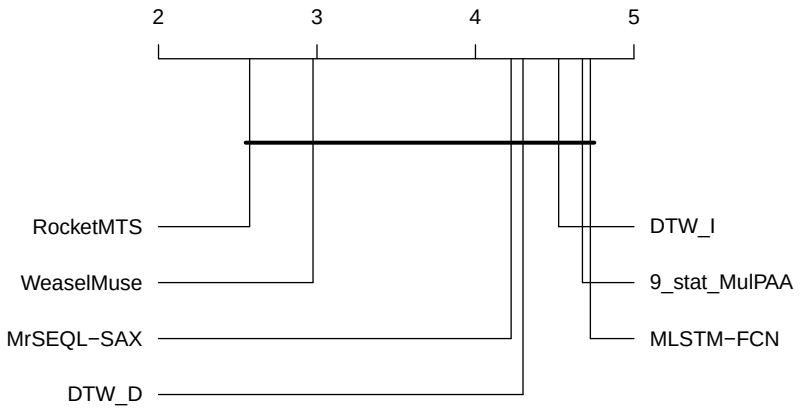


Figure 3.7: Comparison of mean rank classification accuracy for 7 methods across 20 datasets where all methods could deliver results.

Classifiers	Training Time (in minutes)
RocketMTS	18.70
9_stat_MulPAA	30.00
MrSEQL-SAX	157.11
MLSTM-FCN	167.87
Weasel Muse	692.65
DTW_D	3033.79
DTW_I	12587.38

Table 3.5: Runtime of 7 compared methods on 20 datasets.

3.3.2.3 Computation Time

Table 3.5 compares the runtime of the state-of-the-art (SOTA) algorithms, baselines, and our proposed method, 9_stat_MulPAA, on 20 datasets from the UEA archive. The complete runtime results for all datasets are presented in Table 3.3.

Our study revealed that RocketMTS stands out as one of the fastest and most accurate MTSC algorithms, striking a balance between runtime and accuracy. However, it currently lacks compatibility with variable-length time series (Table 3.6). It's worth noting that the results in this table are based on 20 out of 30 datasets that all classifiers could successfully run on.

WEASEL-MUSE, while offering high accuracy, frequently encounters memory issues and is the slowest algorithm among the top performers. MrSEQL shows promise in

[3] https://github.com/sktime/sktime/blob/main/examples/transformation/rocket.ipynb
[4] https://github.com/titu1994/MLSTM-FCN
[5] https://github.com/patrickzib/SFA
[6] https://github.com/alan-turing-institute/sktime
[7] https://github.com/kdd2019-tapnet/tapnet

handling variable-length time series but suffers from high runtime, particularly for lengthy time series. We were unable to reproduce the published results for TapNet.

Our proposed method, 9_stat_MulPAA, demonstrates impressive performance, achieving accuracy comparable to DTW baselines while significantly outperforming them in terms of runtime. Considering its speed advantage, 9_stat_MulPAA emerges as a promising alternative to DTW approaches and warrants further investigation as a potential new baseline.

3.3.2.4 Inter-channel Dependency

The crucial aspect of the MTSC task is the data coming from different channels. In this section, we try to investigate if there is any useful interaction between the different channels of the time series. We chose DTW and ROCKET for this experiment since they support both univariate and multivariate time series. The DTW_I distance is simply the sum of each channel's DTW distance while DTW_D computes the distance in multidimensional space. Intuitively, DTW_D is more capable of capturing the inter-channel dependency in the time series data. Similarly, RocketMTS employs multiple kernels to capture the multidimensional features (one kernel per channel resulting in a random matrix where each row is a kernel corresponding to one channel) while RocketUTS uses kernels that can only capture important information for each channel independently. Surprisingly, we found no significant difference when comparing the two variants for each approach (Figure 3.8). This experiment suggests that dependencies across channels, usually complex and expensive to learn, are perhaps unnecessary, at least with respect to the UCR MTSC Benchmark. A simple linear combination of unidimensional features may already be sufficient for the MTSC task. However, we further explored the interaction among channels in the next chapters in more detail, with respect to memory and computation time.

3.3.2.5 Performance of classifiers on longer time series

A critical aspect of time series classification is the ability of different classifiers to handle longer time series data. In Figure 3.7, we examined the performance of various classifiers on UCR datasets with time series lengths exceeding 500; a total of eight datasets were used. As observed previously, some classifiers are sensitive to time series length, resulting in incomplete results for certain datasets. We compared our proposed 9_stat_MulPAA method to other classifiers on these datasets. We find that RocketMTS

		RocketMTS	MrSEQL-SAX	DTW_I	DTW_D	WEASEL-MUSE	MLSTM FCN
Challenges	Variable-length	C		C	C		C
	Scalability		C	C	C	C	
	Memory Consumption					C	
	Overfitting						C

Table 3.6: Challenges faced by evaluated state-of-the-art MTSC methods. 'C' marks the challenge this method does not yet address well.

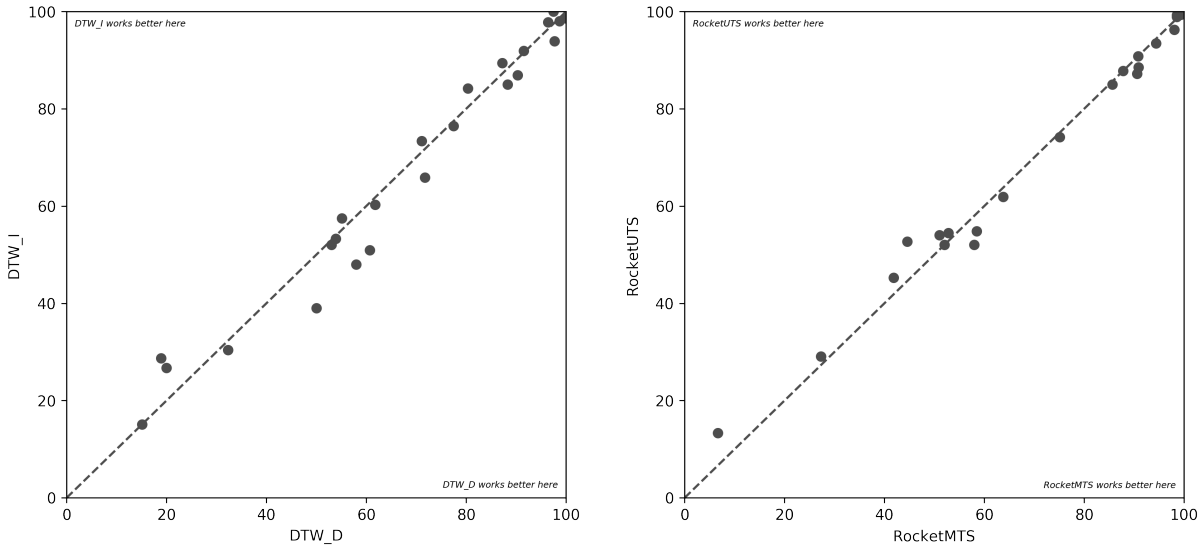


Figure 3.8: Accuracy comparison between (a) DTW_I vs DTW_D and (b) RocketUTS vs RocketMTS. There is no noticeable difference in accuracy between the uni-dimensional and the multi-dimensional variants of the same method.

is able to deal with the longest time series dataset EigenWorms of length 17984 efficiently compared to other classifiers.

3.3.2.6 Comparison of 9_stat_MulPAA with RocketMTS

In Figure 3.9, we compare the performance of 9_stat_MulPAA with the best-performing classifier, RocketMTS, for 26 datasets on which both classifiers were able to run. We found that RocketMTS outperformed 9_stat_MulPAA on 20 of the 26 datasets.

Dataset	TS Length	9_stat_MulPAA	DTW_I	DTW_D	WM	MLSTM-FCN	RocketMTS	MrSEQL-SAX
EW	17984	66.41	60.30	61.80	-	70.99	**90.84**	87.78
MI	3000	47.00	39.00	50.00	-	51.00	**58.00**	51.00
SWJ	2500	33.33	33.30	20.00	26.70	33.33	**53.33**	33.33
EC	1751	28.14	30.40	32.30	47.00	25.10	41.83	**55.51**
CKT	1197	98.61	98.60	**100.00**	98.00	81.94	**100.00**	98.61
SR2	1152	43.89	53.30	**53.90**	52.80	51.67	52.78	50.56
SR1	896	76.79	76.50	77.50	69.60	**87.03**	85.67	68.26
AF	640	33.33	**26.70**	20.00	40.00	26.67	6.67	26.67

Table 3.7: Accuracy comparison of state-of-the-art classifiers on datasets with lengths greater than 500. - represents where the classifier was not able to finish the computation.

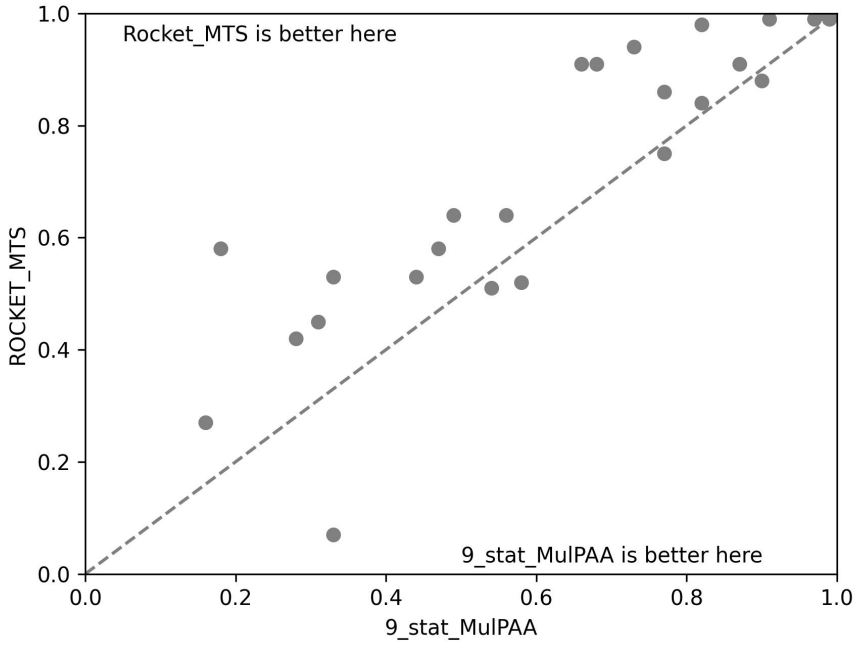


Figure 3.9: Comparison of accuracy of 9_stat_MulPAA with RocketMTS classifier on 26 datasets.

3.4 Conclusion

In this chapter, we have evaluated recent state-of-the-art methods for multivariate time series classification on the UEA MTSC benchmark. We found that, while the methods

compared do not achieve statistically significant difference in accuracy on a subset of 20 datasets where most methods can complete training and prediction, the recent method ROCKET appears to be the most accurate, with the linear classifiers Weasel-Muse and MrSEQL not far behind. Moreover, when considering running time, ROCKET appears to be the most appealing solution for MTSC. We also identified weaknesses in all of the SOTA methods which can hinder their applications to various problems. In particular, most methods do not work on variable-length time series (e.g., ROCKET) and run into memory issues or long runtime if the time series are very long (e.g., Weasel-Muse or MrSEQL). Recent deep learning methods proposed for MTSC, TapNet and MLSTM-FCN, do not achieve higher accuracy than the baselines. These challenges point out worthwhile directions for future work in this area.

In addition, we made a case for an alternate baseline method which is simple, fast to train and as accurate as DTW$_D$ and DTW$_I$. Our approach mainly relies on some well-known statistical properties of time series, hence they are easy to implement and fast to compute. Furthermore, these statistical features can combine nicely with windowing or smoothing techniques (e.g., PAA) to enhance the performance of their respective models. Our best proposed method 9_stats_MulPAA relies only on nine statistical features and multiple levels of PAA smoothing and can perform comparably to DTW methods while running significantly faster.

From our study, we also want to question the necessity of capturing complex channel dependencies in MTS data. Our experiments on two ROCKET and two DTW variants show that this extra knowledge has little impact on the overall accuracy of the models. While this is by no means conclusive evidence, we believe that this question should be considered when working on MTSC. A unified and ongoing comparison of MTSC algorithms is critical to maintain a clear overview of this research area, to identify strengths and weaknesses of existing approaches, and most importantly to facilitate and advance the research in this important topic which has a direct impact on a range of real-world applications.

CHAPTER

FOUR

FAST CHANNEL SELECTION FOR SCALABLE MULTIVARIATE TIME SERIES CLASSIFICATION

4.1 Introduction

In the previous chapter, we discussed the challenges of scalability in multivariate time series classification (MTSC), along with empirical evaluation of state-of-the-art classifiers. We identified that the length of the time series and the number of channels are two major bottlenecks. We proposed a method to deal with the length of time series. In this chapter, we focus on addressing the challenge of channel selection.

Not all channels in an MTSC dataset are equally important for classification. Some channels may be redundant or irrelevant, including which can make the classifier slower and less accurate. Therefore, it is important to select the most informative channels before training the classifier. Any MTSC algorithm should be able to identify and efficiently use the relevant set of channels rather than treating all channels equally.

For example, in Figure 4.1, an MTSC algorithm should be able to identify the upper body parts of the person performing the Military Press exercise, as it targets upper body muscles [62].

This chapter proposes a new method to select relevant channels from the training data before training a classifier. Our primary objective is to enable existing state-of-the-art (SOTA) classifiers to scale better with increased MTSC channels by reducing the time and memory required for computation while maintaining accuracy. In particular, we examine the impact of our channel selection approach on the recent MTSC algorithms Rocket [31], MrSEQL [24], Weasel-MUSE [29] and 1NN-DTW [23].

The main contributions of this chapter are:

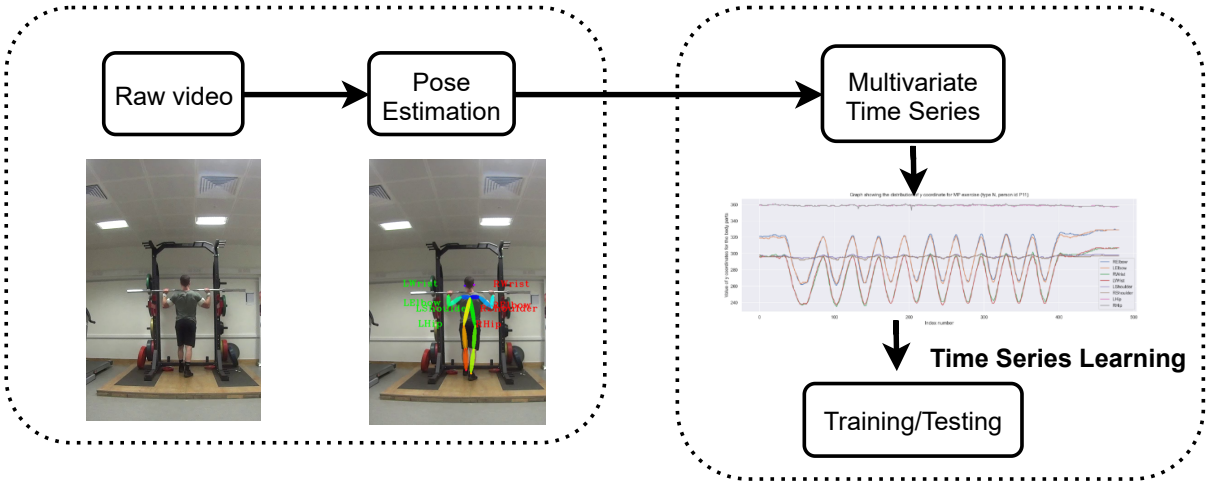


Figure 4.1: From video to multivariate time series using OpenPose (figure from [2]).

- We propose three greedy channel selection strategies for MTSC, to scale up existing MTSC algorithms.
- We conduct extensive experiments on the UCR MTSC benchmark and report a 70% reduction in computation time for the combination of channel selection plus training MTSC algorithms, while preserving the classifier accuracy.
- We show that not all the data is useful for classification and that we achieve significant data storage savings, e.g., 70% of the original data can be discarded with our approaches.
- We present a case study of our methods on a real-world, 25-channel MTSC dataset recorded for the Military Press strength and conditioning exercise.

The rest of the chapter is structured as follows. Section 4.2 presents our proposed methods. Section 4.3 introduces the UEA MTSC benchmark used for the experiments and reports our empirical results. In Section 4.4, we perform a case study on the Military Press dataset. We conclude our study in Section 4.5.

4.2 Methods

Let $X \in R^{n \times d \times l}$ be an MTS dataset and y the labels of the time series in the dataset. We denote by n the number of time series in the dataset, d the number of channels in the

multivariate time series and l the time series length. In this chapter, we only consider fixed-length time series datasets.

The proposed channel selection method uses class centroids as representatives of the classes. Let $X_A = \{t \in X \mid y(t) == A\}$ be the subset of X that contains only samples from class A. The centroid of class A is computed as the average of the time series in that class:

$$C_A[i,j] = \frac{\sum_{k=1}^{k=m} X_A[k,i,j]}{m}$$

where m is the number of samples in class A, the multi-channel centroid C_A is a $d \times l$ matrix in which each row $C_{A,i}$ is the centroid of class A for channel i.

The centroid-driven channel selection technique computes the distance matrix for every pair of class centroids for each channel, using a distance function (Δ) discussed later in the section. For a dataset with r classes, the total number of pairs of class centroids is $\frac{r*(r-1)}{2}$. For instance, the distance matrix for a 4-channel dataset with four classes is shown in Table 4.1, where channel *RElbow* has the highest distance (166.99) for the pair of class centroids C_n and C_r. In our proposed method, we examine this distance matrix to select the channels that are most likely to be useful. The idea is that channels with larger distances between class centroids are more likely to be discriminative since centroids behave like prototypes for time series in those classes. In this example, the distance between class centroids C_n and C_r is highest for channel $RElbow$; therefore, this channel is more likely to be useful in separating these classes than the other channels, while channel $Nose$ has small distances for all class pairs, and so does not seem to be useful to separate any classes.

Our method has three components: a distance measure used to compare centroids, an elbow cut heuristic used to threshold the ranked list of channels and three channel selection strategies.

Table 4.1: Illustration of a distance matrix with 4 channels and 4 classes: $a, arch, n, r$. More details about this dataset are provided in the case study described in Section 4.4.

$Channels$	$\Delta(C_a, C_{arch})$	$\Delta(C_a,C_n)$	$\Delta(C_a,C_r)$	$\Delta(C_{arch},C_n)$	$\Delta(C_{arch},C_r)$	$\Delta(C_n,C_r)$
Nose	15.93	11.13	14.90	14.20	14.60	16.93
RElbow	34.62	48.95	157.12	33.04	153.80	166.99
RHeel	29.66	39.84	9.15	16.65	25.86	35.88
LWrist	38.557	42.95	148.48	47.40	155.56	157.55

Distance Metric. In our current work we use euclidean distance to calculate the distance (Δ) between the centroid pairs for each channel. The Euclidean distance is measured as the $l2$ norm of the difference between the centroids.

$$\Delta(C_{A,i}, C_{B,i}) = \|C_{A,i} - C_{B,i}\|$$

Channel Selection Strategies. We propose and evaluate three different strategies for channel selection to identify useful channels.

- **KMeans.** This strategy applies k-means clustering with $k = 2$ on the distance matrix to segregate the channels. Every channel (row from distance matrix) is assigned to one cluster. The cluster centroid represents the mean distance of channels across every class pair. Thus, the mean of the cluster centroid acts as a discriminating criterion. We select the channels from the cluster whose centroid-mean is greater than the other centroid-mean, meaning that this cluster contains channels with higher separation distance, while the other cluster contains noisy channels.

- **Elbow Class Sum (ECS).** From the distance matrix, we sum all the pairwise distances for each channel (sum each row). The sum of the distances is sorted in descending order, and an elbow-point is retrieved using the elbow-cut approach described below. All the channels with a distance higher than that of the elbow point are selected as the relevant channels. A single large centroid-pair distances can bias this type of channel selection, favouring channels that separate two classes clearly, but may not be useful for separating other classes.

- **Elbow Class Pairwise (ECP).** The second strategy, ECS, can be biased towards channels that are useful for separating only a few classes. An alternative strategy iterates through every class pair, selects the best set of channels for that pair and finally takes the union of channels over all pairs. This eliminates the potential bias found in the previous strategy. In some cases, this can lead to selecting all the channels, however, there were only few instances of this behaviour in the UEA dataset.

Elbow Cut. The elbow cut method [63] is a method to determine a point in a curve where significant change can be observed, e.g., from a steep slope to almost flat curve. This point is often referred to as the elbow or the knee point. This is a well-known method to determine the best number of clusters when doing clustering. We apply it here to separate useful channels from noisy channels. An algorithm takes the sorted

distances corresponding to channels as input and returns the elbow point. The elbow is the point at the highest distance (d) from the line (b) joining the initial and ending point, as shown in Figure 4.2. The distance d to any point on the distance curve is calculated as

$$d = \left| p - (p \cdot \hat{b})\hat{b} \right| \tag{4.1}$$

where $\hat{b} = \frac{b}{\|b\|}$ and $p.\hat{b}$ is the projection of p onto $\hat{b}$. The elbow-point is then elbow $= argmax(d)$. The channels that come before the $elbow$ are selected as useful channels for classification and the smaller dataset with this subset of channels is used for the classification step (see Algorithm 1). It is clear from Figure 4.2 that the elbow point can be relaxed thus allowing a trade-off between data storage and the accuracy of classification. In our work we use the first elbow point which corresponds to channel RShoulder and select only the channels before this point.

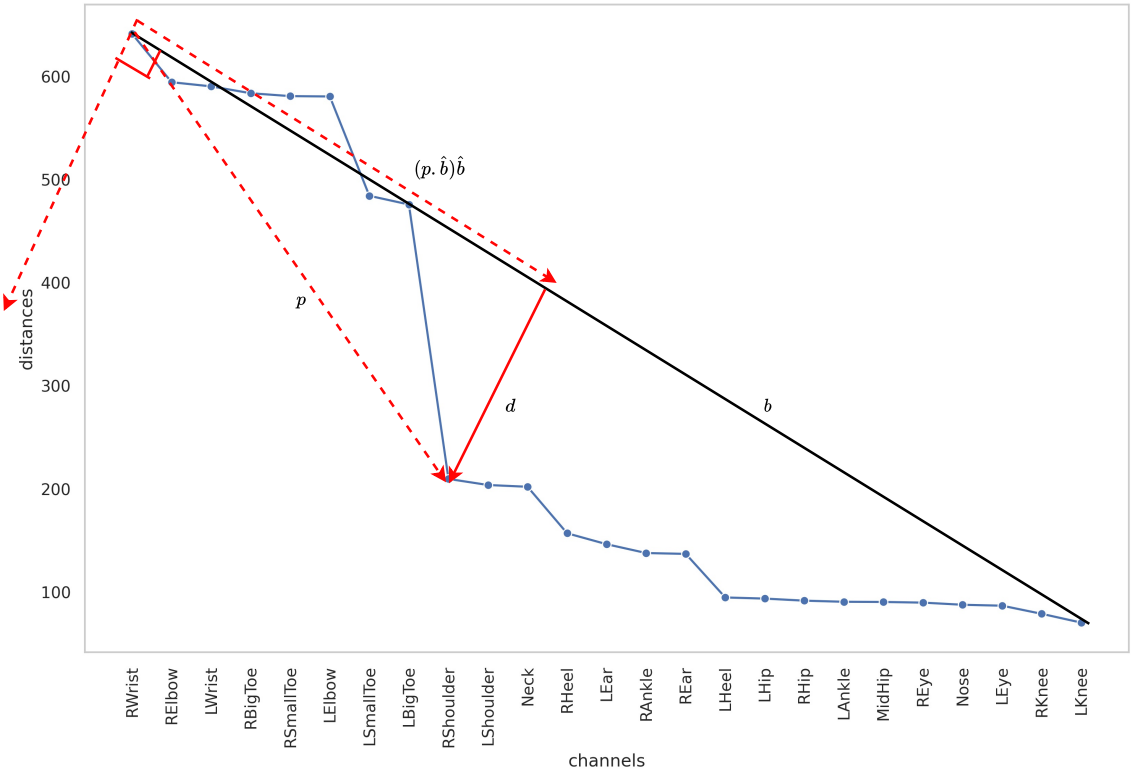


Figure 4.2: Elbow-point channel selection. All the channels up to RShoulder are selected.

Algorithm 1: Channel Selection for an MTSC dataset.

Input: Train dataset: X,y
Output: Selected channels
Initialization;
For each channel in X and each class, compute class centroid ;
Compute distance matrix for all pair of centroids;
if Channel-Selection is KMeans;
    Create 2 clusters using KMeans;
    Selected channels = cluster with higher centroid mean;
elif Channel-Selection is ECS ;
    Sum the distance matrix by rows;
    Rank channels by sum distance;
    Find the elbow on the ranked channels;
    Selected channels = channels with sum distances $>$ elbow point ;
elif Channel-Selection is ECP;
    For each pair of classes;
        Rank the channels by the distances in the corresponding column of the distance matrix;
        Find the elbow on the ranked channels;
        Selected channels = Selected channels $\cup$ channels with class pair-wise distance $>$ elbow point;
Return Selected channels;

4.3 Evaluation

All the experiments were conducted using the popular Python library sktime [64]. Our primary objective in designing experiments is to understand the relative gain or loss in computational aspects of MTSC algorithms using the proposed channel selection strategies for data reduction. We release all our code in our Github repository[1].

4.3.1 Datasets

The UEA/UCR Time Series Classification archive [12] is a collection of univariate and multivariate time series data. The repository contains 30 multivariate datasets from several application domains, e.g., ECG, motion classification, spectra classification. These heterogeneous datasets vary regarding the number of channels (from 2 to 1,345), number of time series (12 to 30,000) and time series length (8 to 17,894). Here we work with a subset of 26 datasets with equal-length time series. Table 4.2 presents the size of each dataset, along with the amount of data reduction achieved using channel selection techniques, which will be discussed later.

[1] https://github.com/mlgig/Channel-Selection-MTSC

Dataset	OriginalSize	KMeansReduced	ECSReduced	ECPReduced	KMeansSaved%	ECSSaved%	ECPSaved%	Channels
DDG	147.25	28.03	40.40	42.37	80.97	72.56	71.23	1345
PSF	315.83	92.81	41.00	104.29	70.61	87.02	66.98	963
FD	511.20	173.95	42.60	42.60	65.97	91.67	91.67	144
MI	409.53	70.39	95.98	95.98	82.81	76.56	76.56	64
Heartbeat	40.06	2.63	5.91	5.91	93.44	85.25	85.25	61
FM	4.52	2.42	0.97	0.97	46.43	78.57	78.57	28
NTP	2.24	1.12	1.12	1.59	50.00	50.00	29.17	24
PS	65.10	11.84	11.84	11.84	81.82	81.82	81.82	11
HMD	5.09	3.56	3.05	4.07	30.00	40.00	20.00	10
AWR	3.04	0.34	1.01	3.04	88.89	66.66	0.00	9
SR2	12.49	5.35	8.92	8.92	57.14	28.57	28.57	7
BM	0.21	0.07	0.07	0.07	66.63	66.63	66.63	6
CKT	6.00	4.00	4.00	6.00	33.33	33.33	0.00	6
EW	105.47	17.58	17.58	70.32	83.33	83.33	33.33	6
LSST	5.97	2.98	2.98	5.97	50.00	50.00	0.00	6
RS	0.32	0.05	0.11	0.22	83.30	66.64	33.32	6
SR1	11.20	5.60	5.60	5.60	50.00	50.00	50.00	6
ER	0.08	0.04	0.02	0.08	49.92	74.88	0.00	4
SWJ	0.92	0.46	0.46	0.46	49.99	49.99	49.99	4
EP	0.70	0.23	0.23	0.47	66.66	66.66	33.33	3
EC	10.56	7.04	3.52	10.56	33.33	66.67	0.00	3
HW	0.58	0.39	0.19	0.58	33.33	66.65	0.00	3
UW	0.91	0.61	0.30	0.91	33.33	66.66	0.00	3
AF	0.15	0.08	0.08	0.08	49.96	49.96	49.96	2
LB	0.17	0.09	0.09	0.17	49.96	49.96	0.00	2
PD	2.86	1.43	1.43	2.86	50.00	50.00	0.00	2
Memory (GB)	**1.62**	**0.42**	**0.28**	**0.42**				

Table 4.2: The amount of memory (MB) used by each dataset when using all channels and after applying our channel selection strategies.

4.3.2 MTSC Algorithms

All algorithms described in Chapter 2, except ROCKET, utilise all the channels from the MTSC datasets. Table 4.3 gives the hyperparameter settings used for all the classifiers in this study.

Table 4.3: Hyperparameter setting used for various SOTA methods

Classifiers	Hyperparameter-setting
WEASEL-MUSE	MUSE(random_state=0)
MrSEQL-SAX	MrSEQLClassifier(seql_mode=fs, symrep= ["sax"])
ROCKET	Rocket(random_state=0)
ROCKET*	ROCKET with all channels
1NN-DTW	KNeighborsTimeSeriesClassifier(n_neighbors=1, distance="dtw")

Table 4.4: Mean Accuracy and total time of SOTA on 26 UEA MTSC datasets before channel selection.

Classifier	Accuracy	Time(in hrs)
ROCKET	71.59	0.1
WEASEL-MUSE	70.28	73.22
MrSEQL-SAX	66.99	141.40
1NN-DTW	65.38	152.07

Table 4.4 presents the results of these MTSC algorithms on the 26 datasets when no explicit channel selection is implemented. The time is shown in hours and is the total time taken by the algorithm for training and prediction. We observe that ROCKET and WEASEL-MUSE have almost similar mean accuracy, however, ROCKET is much faster. ROCKET implements a random channel selection strategy which allows it to keep the runtime bounded, no matter how many channels the dataset has; we discuss this in more detail later in this section. ROCKET also uses a multi-threaded implementation, while all the other algorithms are single-thread implementations, hence the significant difference in runtime. The baseline 1NN-DTW is the least accurate and the slowest method among the four. MrSEQL-SAX does not use the SFA representations in this chapter due to the SFA implementation in sktime being too computationally expensive, its accuracy is lower than ROCKET and WEASEL-MUSE in this experiment. Both WEASEL-MUSE and MrSEQL are impacted by the runtime taken by the symbolic transform, while the DTW computation impacts 1NN-DTW.

4.3.3 MTSC with Channel Selection

Our proposed channel selection strategies (KMeans, ECS, and ECP) are evaluated on the same 26 datasets as above. The channel selection algorithm is run before the MTSC algorithm and it typically results in a reduced dataset for training/testing. We investigate how these strategies impact the classification accuracy, running time (training and testing) and data storage size.

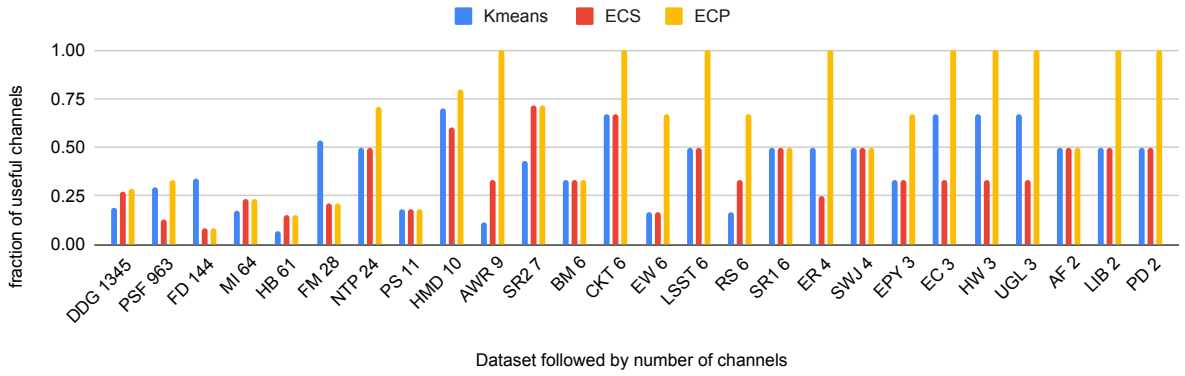


Figure 4.3: Fraction of channels selected by each of three channel selection strategies.

Ratio of Channels Selected. Figure 4.3 illustrates the proportion of channels selected by our methods for each dataset, with 1.0 representing that no channels were discarded. The figure on the x-axis presents the different datasets organized in descending order according to the number of channels, while the y-axis displays the number of selected channels. For each dataset, we differentiate the number of channels selected by employing different colour coding for various channel selection methods. The acronyms and details of the datasets can be found in the Table 2.1. The ECP ratios appear to be higher in general, as expected and mentioned in Section 4.2. However, this mostly occurs in datasets with few channels (the right side of Figure 4.3). On the left side, where the large numbers of channels can become an issue, ECP appears to be just as efficient as the other methods. We also observe that all three methods are more effective on datasets with a larger number of channels that usually pose a significant scalability challenge to existing MTSC algorithms.

In Table 4.5, we show the total time taken by the three-channel selection strategies. This includes the time each method takes to compute centroids and create the distance matrix. All three techniques are run only on the training dataset, and the output is a selected subset of channels. Since these methods only require the distance matrix for centroid pairs, they are extremely fast even for large datasets, as the time complexity is only affected by the number of classes, and not by the number of samples. The subset of selected channels is then used to create a reduced dataset as input to MTSC algorithms.

Table 4.5: Total time taken by three channel selection strategies on 26 UEA datasets.

Channel Selection Strategy	KMeans	ECS	ECP
Total Time (minutes)	0.34	0.33	0.35

Table 4.6: Loss/Gain in mean accuracy (ΔAcc) vs percentage time saved (%Time) with respect to All channels (Table 4.4) for our three channel selection techniques on 26 UCR datasets. The red and blue color indicates loss and gain in accuracy respectively. Higher value for %Time or %Storage indicates more time or storage saved.

Channel Selection→ Classifiers↓	KMeans ΔAcc | %Time	ECS ΔAcc|%Time	ECP ΔAcc | %Time
ROCKET	-4.01 | 33.62	-4.40 | 29.23	+0.13 | 21.43
WEASEL-MUSE	-4.63 | 70.46	-3.80 | 79.90	-1.53 | 73.21
MrSEQL-SAX	-3.33 | 72.68	-3.80 | 84.00	+0.45 | 77.06
1NN-DTW	-4.28 | 68.30	-6.08 | 68.82	+0.67 | 44.80
Mean ΔAcc| Mean %Time	-4.06 | 61.26	-4.52| 65.48	-0.07 | 54.12
Mean %Storage Saved	73.95%	82.59%	74.38%

Performance of Channel Selection. Table 4.6 shows the change in accuracy and the percentage of time saved by the MTSC algorithms when run on the reduced datasets after applying the three channels selection strategies.

The comparison reveals that there is a massive gain in computation time for a minimal drop in accuracy. The time taken to find the subset (Table 4.5) is insignificant in comparison. Out of the three channels selection strategies, ECP seems to be the best choice for channel selection. It significantly reduces the computation time and at the same time eliminates noisy channels, thus increasing the accuracy for ROCKET, 1NN-DTW and MrSEQL-SAX. The method WEASEL-MUSE takes a small hit on accuracy (1.5%), at the benefit of saving 73.21% in runtime. Considering that WEASEL-MUSE requires 73.2h to complete training and prediction on this benchmark (see Table 4.4), this is a significant time saving. A similar result holds across all classifiers, and all channel selection strategies: for a small loss in accuracy, there is a high gain in runtime. In the case of ECP, the accuracy is preserved or even increased, with a significant saving in runtime. We also calculate the average amount of memory saved by the channel selection techniques over the 26 datasets. The comparison of dataframe size in memory, before and after channel selection is used to compute these values. Overall, this MTSC archive uses about 1.6 GB of memory, as shown in Table 4.2, and when using our channel selection strategies, this is reduced to less than 30% of the original size. When stored on disk this dataset is about 3.3 GB total, and with the channel selection techniques, this is reduced to about 900MB.

4.3.4 Effectiveness of Channel Selection

In this experiment we test whether our best strategy (ECP) selects useful channels and how good the selection is compared to selecting optimal channel subsets.

Optimal Channel Subset Selection. We evaluate every possible subset of channels on the test set to discover the optimal subset. Naturally, this brute-force approach is very expensive and impractical for datasets with a high number of channels as the possible combination for a dataset with d channels will be $2^d - 1$. Nevertheless, in this chapter, all the subsets for datasets with a number of channels <4 are analysed. These datasets are: *AtrialFibrillation*, *Libras*, *PenDigits*, *EthanolConcentration*, *Epilepsy*, *Handwriting*, *UWaveGestureLibrary*. In this experiment, we choose the state-of-the-art ROCKET classifier to quickly evaluate all the subsets. However, because ROCKET internally randomly samples the channels, it can select a good subset by chance and mask the issue of selecting bad channels. Therefore, we modify its code to get ROCKET to use all channels in each kernel, i.e., we use the ROCKET* variant. By doing so, the impact of a good channel subset and a bad channel subset on classification accuracy becomes more pronounced.

Table 4.7: Accuracy of ROCKET* on datasets with channels < 4. Bold indicates the optimal subset. Underscore indicates the subset selected by ECP. Empty spaces are for datasets with fewer channels, e.g., dataset AF only has two channels, 0 and 1.

DT	0	1	2	(0,1)	(0,2)	(1,2)	(0,1,2)
AF	**20**	6.67		13.33			
LB	73.9	77.78		**93.89**			
PD	89.59	88.45		**98.26**			
EC	54.4	49.8	**53.6**	38.0	44.9	39.2	36.1
EP	97.82	**100**	94.93	98.55	98.55	97.83	99.28
HW	38.12	32.35	42.12	45.76	59.76	50.35	**57.06**
UW	79.37	71.25	71.88	87.5	93.12	84.06	**93.75**

Table 4.7 shows that ECP successfully identified the optimal subset five out of seven times. With the Epilepsy (EP) problem, it also correctly identified channel 0 as a potential issue (the classification accuracy is only 97.82% with channel 0 alone) and excluded it from the selection. However, for this dataset it seems to be better to use either only channel 1 or all the channels. It is important to remind the reader that this setting is evaluated directly on the test data, and in practice we do not have perfect knowledge of the best subset of channels for the test data. ECP selects this channels based on the training data alone, and it seems to be effective at finding the useful channels for each task using only training data.

Random Channel Subset Selection. In order to further understand the effect of the ECP channel selection method, we compare the accuracy of the ROCKET classifier, when using channels selected with different strategies. We compare ECPRocket (ECP combined with ROCKET) with ECPsizeRandomRocket, a simple baseline where the

number of channels is set using ECP, but the actual channels are picked randomly. We repeated the experiment 10 times for each dataset and report the average accuracy in Figure 4.4. We observe that for the majority of the large datasets (number of channels > 10), ECPRocket is better, while for datasets with less number of channels (number of channels ≤ 10) the ECPSizeRandomRocket works similar to ECPRocket. Note that for half of the datasets with number of channels ≤ 10, ECP does not reduce the number of channels (i.e., it keeps all the channels as shown in Figure 4.3), hence the two variants ECPRocket and ECPSizeRandomRocket simply reduce to ROCKET, since ECP has no effect in this case. For datasets with a higher number of channels, ECP often reduces the full channel set to a subset of good channels, and the variant ECPRocket constrains ROCKET to work with this pool of good channels, resulting in storage savings and improvements in accuracy. Hence, for either small or large number of channels, ECP is fast and leads to storage savings without resulting in loss of accuracy.

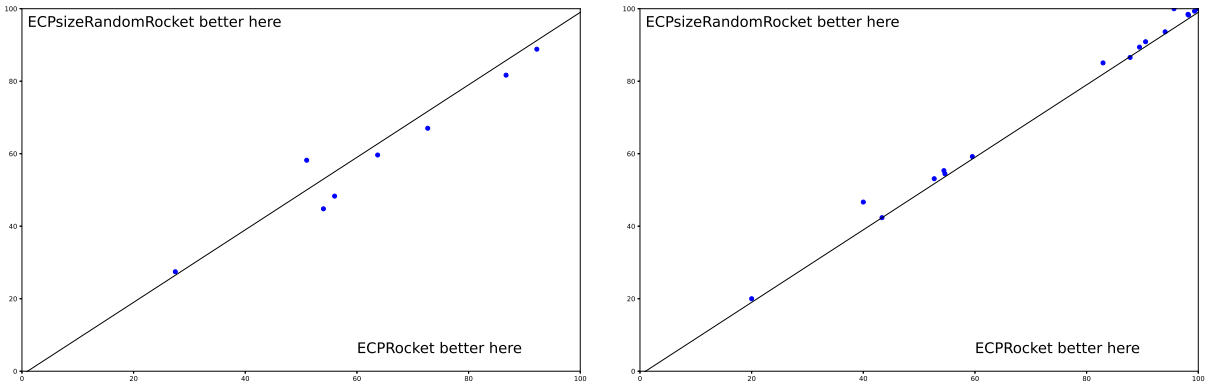


Figure 4.4: Comparison of ECPRocket with ECPsizeRandomRocket. Figure (a) represents datasets with the number of channels > 10, and Figure (b) represents datasets with the number of channels $<=10$.

4.4 Case Study: Channel Selection for the Military Press MTSC Dataset

4.4.1 Dataset

A total of 56 healthy volunteers (34 males and 22 females; age: 26 ± 5 years, height: 1.73 ± 0.09 m and body mass: 72 ± 15 kg) participated in a study aimed at analysing the execution of the Military Press strength and conditioning exercise. The participants

Table 4.8: Channel Selection using our three strategies. All strategies select the same 8 body parts as relevant for this classification task.

Channel Selection	Body Parts
KMeans	Elbows, Wrists, BigToes, SmallToes
ECS	Wrists, Elbows, BigToes, SmallToes
ECP	Elbows, Wrists, BigToes, SmallToes

completed ten repetitions of the normal form and ten repetitions of induced forms. The NSCA guidelines were applied under the guidance of sports physiotherapists and conditioning coaches to ensure standardisation. The dataset was extracted from the video of individuals performing the exercise with the help of the human body pose estimation OpenPose[2]. There are four classes in the dataset, namely: Normal (N), Asymmetrical (A), Reduced Range (R) and Arch (arch). The N refers to the correct execution of the exercise; A refers to when the barbell is lopsided and asymmetrical, R refers to the form where the bar is not brought down completely to the shoulder level and Arch refers to when participants arch their back. A total of 25 body parts were tracked, as seen in Figure 4.2. These 25 body parts act as channels for the MTSC task. The train and test size for this dataset is 1452 and 601 respectively and the length of time-series is 160.

4.4.2 Channel Selection

Table 4.8 illustrates the selected channels for the Military Press dataset. The Elbows and Wrists are actively involved in the exercise, as the participant is required to lift a barbell over the shoulders. However, the Toes do not seem to contribute to the exercise. We tried to investigate this and think that the issue might be related to data pre-processing when the time series is extracted from the video; investigating this aspect further is interesting but outside the scope of this theisis.

4.4.3 Results & Discussion

Table 4.9 reports the results for ECP with different SOTA MTSC classifiers. ROCKET is the fastest and most accurate classifier in this experiment. The data normalisation which is turned on by default in ROCKET, is turned off in the current experiment. This is due to the fact that the signal magnitude contains important information for this task, so normalisation should not be used in this case. For WEASEL-MUSE and

[2] https://github.com/CMU-Perceptual-Computing-Lab/openpose

Table 4.9: Performance of ECP on the Military Press Exercise.

	Accuracy	Time (minutes)
Classifiers	ECP | All	ECP| All
ROCKET	76.26 | 77.53	2.14 | 2.25
WEASEL-MUSE	57.57 | 57.57	30.29 | 107.02
MrSEQL-SAX	58.23 | 61.56	139.53 | 516.79
1NN-DTW	48.58 | 47.25	10.39 | 24.36
Data Size (MB) Reduced|Original	15.77 | 49.29	

MrSEQL-SAX, data normalisation is done internally in the algorithm during the symbolic transform (SFA/SAX), so we cannot de-activate the data normalisation step. This affects the accuracy of these methods in this task, since the magnitude of the signal is important to differentiate between classes. As in the previous experiments, in the case study we also find that ECP saves a large amount of time and memory, with minimal or no loss in accuracy. For WEASEL-MUSE, it saves about 71.6% of computation time, while for MrSEQL-SAX and 1NN-DTW it saved about 74% and 68%, respectively. Moreover, the memory required for computation is reduced to 32%, thus a saving of 68% on the original dataset.

4.5 Conclusion

In this chapter, we have shown that not all the channels for MTSC are helpful. Data noise in the form of uninformative channels can prevent the classifier from achieving its maximum potential. We have observed that channel selection can remove some of the noise and drastically reduce the required computation time for existing MTSC methods. In the current chapter, we showed that the distance between the class centroids of various channels plays a crucial role in identifying the noisy channels. Our three-channel selection strategies ECP, ECS and Kmeans, can select the useful channels based on this distance. All three techniques significantly reduced the runtime and memory required to run SOTA classifiers. The ECS and KMeans techniques also reduced the accuracy, while ECP resulted in accuracy gains for MrSEQL-SAX, ROCKET and 1NN-DTW and marginal accuracy loss for WEASEL-MUSE. We believe that with a more robust elbow selection heuristic the performance can be improved further. Our channel selection techniques significantly reduced the data size on disk for most of the MTSC datasets, thus enabling significant storage savings for large MTSC datasets where several channels are not useful for the classification task.

One significant limitation of the channel selection methods discussed in this chapter is their vulnerability to noise interference. When the data set contains outliers, these outliers can impact the class prototypes significantly. This impact can be detrimental, as it may distort the overall representation of the class and lead to inaccurate channel selection and, consequently, wrong classifications. In the next chapter, we address this issue in detail by proposing robust channel selection methods.

CHAPTER

FIVE

SCALABLE CLASSIFIER-AGNOSTIC CHANNEL SELECTION FOR MULTIVARIATE TIME SERIES CLASSIFICATION

5.1 Introduction

In our previous discussion, we explored techniques for channel selection, which effectively enhances the scalability of classifiers. The findings of [52] corroborate the effectiveness of channel selection methods introduced in the preceding chapter. The authors employed ECS and ECP techniques to scale their proposed ensemble HIVE-COTEv2 classifier. Their results indicate that utilizing ECP as a channel selection technique with HIVE-COTEv2 yields improved efficiency without loss in accuracy to the default HIVE-COTEv2 model, surpassing other channel selection and dimension reduction strategies.

However, the centroid method proposed in the last chapter for creating class prototypes is affected by noise. In this chapter, we propose a broader and more robust view of creating class prototypes, which extends the previous channel selection methodology by studying different types of prototypes for representing classes, refining our channel selection and ranking methods, and studying robustness to common data challenges such as noise, and minority classes.

We evaluate our methods on the UEA MTSC benchmark and on synthetic time series, which simulate different types of signals, such as Gaussian process (GP), pseudo-periodic (PP), and autoregressive (AR), where the relevant signal is either static or moving. These simulated datasets help us to analyze the behaviour of channel se-

lection across different domains, such as EEG waveforms (generally pseudo-periodic [65]), finance data (usually follows auto-regressive patterns [66]), and vibration mechanics (involves a lot of Gaussian processes [67]).

We also include a real-world case study on a human motion dataset collected as a multivariate time series with 50 channels. We encounter challenges such as input noise due to the data collection process, minority classes, and relevant patterns that can shift for different time series due to the different motion characteristics of different people.

The primary objective of this chapter remains the same as the last chapter, which is to perform fast channel selection, leading to a reduction in data size and computation time without compromising accuracy. We aim to achieve this by removing unnecessary channels so that existing MTSC algorithms can perform classification efficiently and accurately.

The main contributions of this chapter are:

- We present new channel selection techniques, which are fast, scalable, robust to different types of noise in the input data and classifier-agnostic.
- We conduct extensive experiments on the UEA MTSC benchmark and report 75% reduction in computation time and a data reduction of 60% using our approaches, while preserving accuracy.
- We demonstrate a significant improvement in the accuracy (more than 20%) of the state-of-the-art classifier ROCKET on synthetic datasets with varying levels of difficulty in identifying important channels.
- We show significant accuracy improvement for ROCKET (more than 5%) on a real-world case study of our methods on a human motion MTSC dataset with 50 channels, recorded for classifying the Military Press strength and conditioning exercise.

5.2 Methods

In this section, we present our generic methodology for channel selection, which we will demonstrate using an example from our case study. Although we explained the concept in the previous chapter, we will briefly reiterate it here.

The proposed channel selection techniques depend on three components: *class prototype*, *distance matrix* and *distance aggregation* (Figure 5.1). Class prototypes are basically

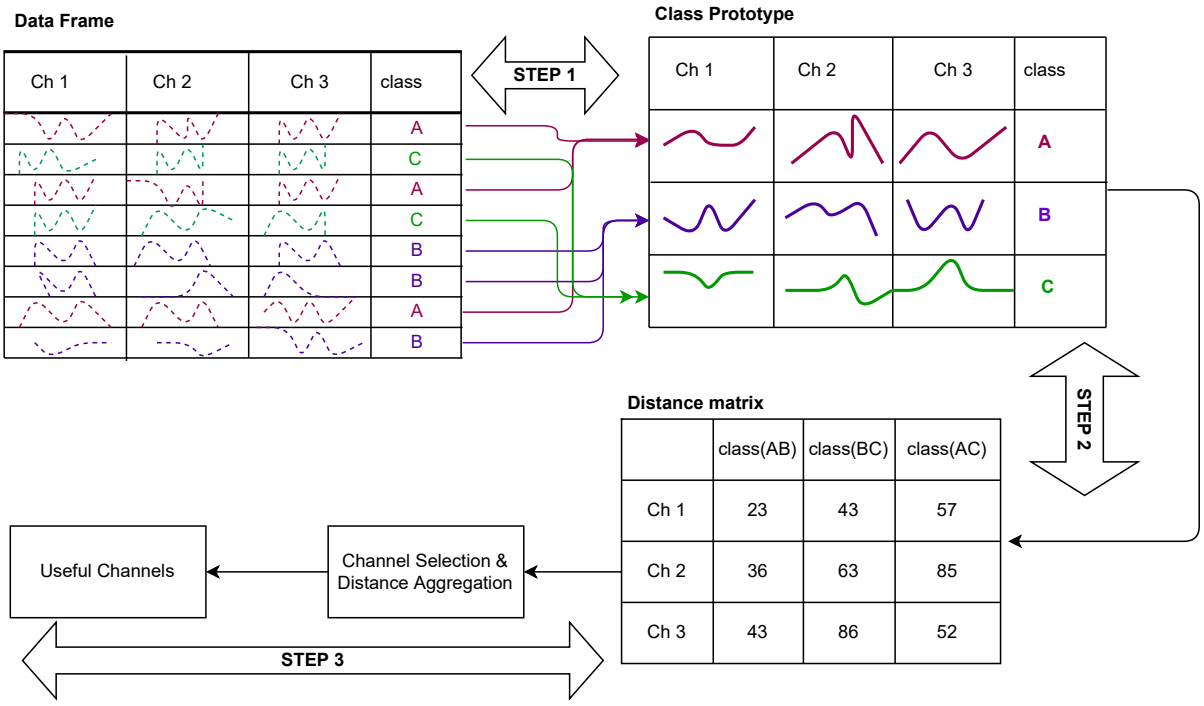


Figure 5.1: Channel Selection Pipeline: (Step 1) Compute class prototypes for each class and each channel. (Step 2) For each channel, compute distance between prototype pairs. (Step 3) Select the channel subset with the highest distance between prototypes. Hypothesis: Channels with higher distance are more useful for separating the classes in the subsequent classification task.

representations of each class in the dataset. The class prototype is also a multivariate time series. The key idea of this approach is that by comparing class prototypes for each channel, we can get an indication of which channel is more useful. The channels that increase the distance between class prototypes are more likely to be useful for the classification task, since they add in class discrimination. In the next step we compute a distance matrix between prototypes (pairwise distances between class prototypes) and use distance aggregation to rank and select channels. Next we discuss different choices of class prototypes and how to employ the distance matrix to select useful channels.

5.2.1 Class Prototypes

A representative prototype of a class is necessary for the proposed techniques. In our approach, it is also important that the class prototypes are inexpensive to compute. Extending our previous work [17] on channel selection, here we investigate other choices for the class prototypes, with a specific focus on robustness to noise. We focus on three prototypes: mean (or centroid as discussed in previous chapter), median and MAD

(data trimmed using median absolute deviation [68]). Next, we formally define the class prototypes.

Let $X \in R^{n \times d \times l}$ be an MTS dataset and y the labels of the time series in the dataset; n represents the number of time series in the dataset, d represents the number of channels in the multivariate time series, and l is the length of the time series.

Mean Prototype: Let $X_A = \{s \in X \mid y(s) = A\}$ be the subset of X that contains only samples from class A. The mean prototype of class A is computed as the *mean* of all the time series in that class, i.e., the centroid:

$$C_A[i,j] = \frac{\sum_{k=1}^{k=m} X_A[k,i,j]}{m} \tag{5.1}$$

where m is the number of samples in class A. The multi-channel mean prototype C_A is a $d \times l$ matrix in which each row $C_{A,i}$ is the centroid of class A for channel i.

Median Prototype: Mean estimation is known to be susceptible to outliers. Median is an alternative to remedy this issue. For class A, the median prototype is computed as:

$$M_A[i,j] = calculate\_median(X_A[k,i,j]_{k=1}^{k=m}) \tag{5.2}$$

where m is the number of samples in class A. The multi-channel median prototype M_A is a $d \times l$ matrix in which each row $M_{A,i}$ is the median-center of class A for channel i. From hereon, we refer to the median prototype as the median for simplicity.

MAD prototype: Another method to handle outliers is clamping, i.e., replacing outliers with predetermined threshold values. We propose the MAD prototype for multivariate time series, which clamps the outliers using Median Absolute Deviation ([68]). The MAD of class A is computed as follows:

$$mad_A[i,j] = median(|X_A[k,i,j] - M_A[i,j]|_{k=1}^{k=m}) \tag{5.3}$$

The upper and lower thresholds for clamping are then defined as

$$\textit{upper\_limit} = M_A + 0.5 * mad_A$$

$$\textit{lower\_limit} = M_A - 0.5 * mad_A$$

For a channel i, the MAD prototype for class A is calculated as the mean of the clamped dataset $\overline{X}$ (values outside of the $[\textit{lower\_limit}, \textit{upper\_limit}]$ range are replaced with the respective boundary values).

$$\text{MAD}_A[i,j] = \frac{\sum_{k=1}^{k=m} \overline{X}_A[k,i,j]}{m} \tag{5.4}$$

The multi-channel MAD prototype MAD_A is a $d \times l$ matrix in which each row $MAD_{A,i}$ is the (clamped) centroid of class A for channel i. For the purpose of simplicity we refer to this prototype as MAD.

Figure 5.2 illustrates the prototypes for two classes in the Military Press dataset. We notice that the mean prototype is more volatile than the other two, which can be explained by the outliers in the data. For this task, the time series data is obtained using body pose estimation libraries, and when the body is not fully visible or the estimation is not accurate, there can be outliers in the data. On the other hand, median and MAD prototypes follow each other closely. However, the MAD prototype seems to be more reflective of changes in the data.

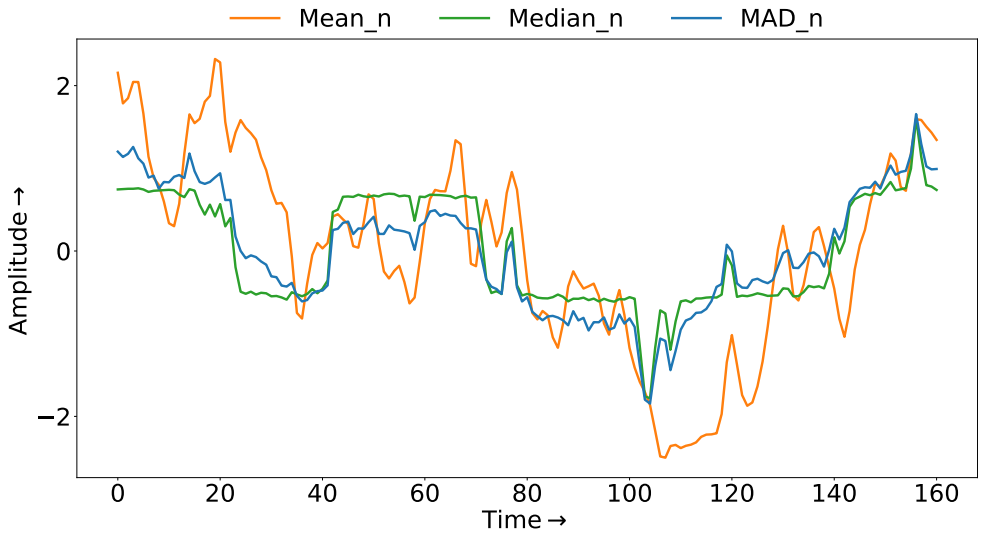
5.2.2 Distance Matrix

We compare the class prototypes by measuring their Euclidean distance for each channel and for each pair of classes. The result is a distance matrix, which is a two-dimensional array of channels and class pairs. In Figure 5.1, the distance matrix tells us that the (Euclidean) distances between class A and class B prototypes in channel 1, 2, and 3 are 23, 36, and 43, respectively. This implies that channel 3 might be more useful than channel 1 and 2 in separating class A and class B. Table 5.1 illustrates a real distance matrix for the Military Press dataset. It is important to note that we also tested DTW as an alternative distance measure, which is usually preferred in time series analysis. Nevertheless, DTW is more expensive, hyperparameter sensitive, and its benefit when compared to Euclidean distance was unnoticeable in our experiments (Tables 5.5 and 5.6).

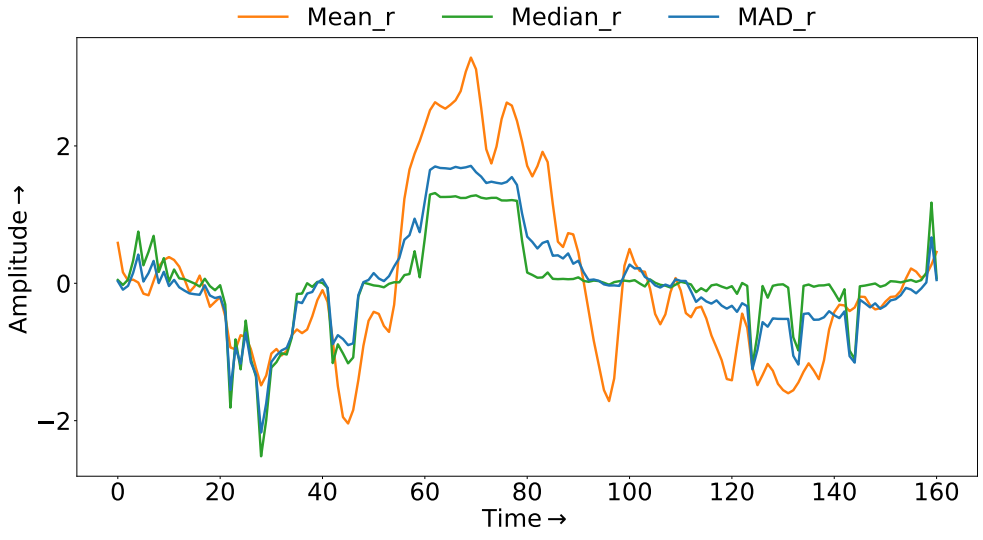
5.2.3 Distance Aggregation and Channel Selection

5.2.3.1 The ECS and ECP Channel Selection Algorithms

The preliminary study [17] proposed three techniques to perform channel selection using a simple class centroid as prototype, namely, KMeans, ECS (Elbow Class Sum), and ECP (Elbow Class Pairwise). Since KMeans performed poorly in comparison with ECS and ECP, we did not include it in this chapter. Here, we present the pseudo-code for our two algorithms ECS and ECP and present new extensions of our methodology in



(a)



(b)

Figure 5.2: Class prototypes for the classes normal (n) in figure (a) and reduced (r) in figure (b) for the left wrist channel along the X-axis (details of this dataset are given in the case study Section 5.6).

Channels ↓ | CP →	← Classes → a_arch	a_n	a_r	arch_n	arch_r	n_r
RWrist_Y	51.00	56.27	103.15	48.93	123.44	82.84
RElbow_Y	50.66	51.67	98.55	35.13	87.03	75.97
LWrist_Y	40.12	45.61	99.03	45.41	123.17	90.40
REar_Y	39.44	5.29	11.39	36.79	31.18	9.85
LEar_Y	38.68	3.26	4.04	39.84	37.40	5.99
RShoulder_Y	35.43	11.63	23.36	34.42	18.33	23.77
RElbow_X	30.66	43.73	76.51	30.60	70.06	68.05
RHip_Y	28.88	6.41	11.38	27.95	19.21	10.52
MidHip_Y	28.23	8.10	9.16	25.60	21.45	7.49
LElbow_Y	27.72	40.91	95.98	33.26	92.82	83.06
LElbow_X	27.51	40.61	55.97	28.79	68.01	68.89
LShoulder_Y	27.22	16.11	10.78	37.82	25.27	20.57
Neck_Y	25.91	14.47	15.41	36.49	18.38	25.95
RSmallToe_X	25.09	24.26	29.78	32.07	33.48	36.95
RWrist_X	24.63	19.88	17.88	14.93	29.84	28.10
LHip_Y	23.97	9.02	7.78	23.49	19.12	8.11
RBigToe_Y	20.66	20.10	16.96	29.57	25.34	30.02
RSmallToe_Y	20.05	14.86	12.26	26.29	19.48	20.93
Neck_X	17.27	6.37	20.87	15.78	26.78	22.75
MidHip_X	16.87	10.10	11.46	13.77	15.98	9.30
LBigToe_X	13.31	6.06	5.71	13.76	11.98	6.10
LWrist_X	13.27	10.90	14.93	9.98	15.47	10.37
LKnee_X	11.57	7.31	9.04	11.69	14.93	8.73
LAnkle_X	11.02	6.89	10.52	10.02	15.16	12.29
RBigToe_X	10.45	6.22	5.57	11.79	11.26	8.36
RHeel_Y	10.40	1.90	4.40	10.55	12.48	4.58
LSmallToe_Y	10.10	7.58	11.15	9.87	10.42	10.03
LHip_X	9.59	7.26	6.46	10.97	13.94	9.65
RKnee_X	9.54	8.88	6.95	12.40	11.68	7.27
REar_X	8.92	16.58	7.71	16.50	7.16	14.55
LHeel_Y	8.81	5.90	6.90	6.53	10.84	6.74
LKnee_Y	7.69	6.18	6.07	6.77	7.70	4.85
RAnkle_Y	7.60	1.19	6.49	7.36	9.31	6.41
LEar_X	7.09	2.14	3.22	6.27	6.74	2.59
RShoulder_X	6.87	10.69	6.45	10.55	7.39	12.42
LBigToe_Y	6.85	6.74	6.54	5.68	5.50	3.37
RHip_X	6.43	7.97	6.47	3.46	1.20	3.89
LShoulder_X	6.38	20.07	11.29	20.81	13.67	20.76
RKnee_Y	6.20	5.50	6.16	4.06	5.35	4.75
LSmallToe_X	5.89	27.39	7.29	29.62	7.87	28.84
RAnkle_X	4.28	2.47	0.15	5.06	4.28	2.47
RHeel_X	3.99	2.20	0.12	4.90	4.02	2.19
LAnkle_Y	0.44	1.78	3.62	1.75	3.66	4.03
LHeel_X	0.16	16.05	13.26	16.08	13.29	17.56
Nose_X	0.00	0.00	0.00	0.00	0.00	0.00
REye_X	0.00	0.00	0.00	0.00	0.00	0.00
LEye_X	0.00	0.00	0.00	0.00	0.00	0.00
Nose_Y	0.00	0.00	0.00	0.00	0.00	0.00
REye_Y	0.00	0.00	0.00	0.00	0.00	0.00
LEye_Y	0.00	0.00	0.00	0.00	0.00	0.00

Table 5.1: Distance matrix for the Median class-prototype on the MP dataset with 50 channels.

order to be more robust to data noise. In particular, we investigate new class prototypes, different ways to compute and aggregate the distance matrix and expand the channel selection and ranking strategy.

Algorithm 2: ECS Channel Selection for an MTSC dataset.

Input: Training dataset: X,y
Output: Selected channels S, ranking of channels, R

Initialization
for *each channel* x_i *in* X **do**
  for *each class* y_j **do**
    Compute the class prototype C as Mean, Median or MAD
  end
end
for *each channel* c_i *in* C **do**
  for *each class* y_j **do**
    Compute the distance matrix M
  end
end

$$\text{Compute distance } D = \sum_{i=1}^{n} M_{ij} \quad \forall \text{class pair}$$

Sort channels by sum in decreasing order
eb = elbow point(sorted channels)
S = channels with sum $>$ eb
R = Rank channels based on total distance summed over class pairs
Return S, R

Algorithms 2 and 3 describe the distance aggregation and channel selection steps for the ECS and ECP methods, respectively. ECS evaluates each channel by the total sum of pairwise distances between class prototypes for that channel. However, for some tasks, some classes are more distinguishable than others; hence, the distance-sum can be dominated by the distances between these classes. As a result, channels that are specifically more useful for the less distinguishable classes might be ignored. To address this issue, ECP iterates through every class pair, selects the best channels by elbow cut for each pair and later does union operation on selected channels. The final set of selected channels is a union over the selection for each pair. In this way, ECP ensures that the channels that might be ignored in ECS, also have a good chance to be selected.

In addition, we have evaluated a simpler baseline technique to rank channels using their *l2-norm*. The basic idea of using the l_2-norm is that channels with higher magnitude might be more useful for discriminating between classes.

Algorithm 3: ECP Channel Selection for an MTSC dataset.

Input: Training dataset: X,y
Output: Selected channels, ranking of channels

```
Initialization
for each channel x_i in X do
    for each class y_j do
        Compute the class prototype C as Mean, Median or MAD
    end
end
for each class y_j do
    Compute the distance matrix M
end
for each class pair in M do
    r = Rank channels in decreasing order of the distance
    eb = elbow_point(r)
    S = S ∪ channels with sum > eb
end
R = Rank channels by total distance summed over class pairs;
Return S, R ;
```

5.2.3.2 Elbow Cut on Ranked Channels

Both the ECS and ECP algorithms employ an heuristic for automatically selecting a set of channels from a given ranked list of channels. This heuristic is commonly known as the elbow cut. The elbow cut selects channels from the distance matrix (ECP) or the distance sums (ECS). It identifies an inflection point on a ranked list of distances (Figure 5.3) and returns the channels based on this point (commonly known as the *elbow point*).

The elbow point (see Figure 5.3) is the point at the highest distance d from the line b joining the initial and ending point in a ranked list. The distance d to any point on the distance curve is calculated by using a vector projection of p on the line b as shown in Equation 5.5.

$$d = \left| p - (p.\hat{b})\hat{b} \right| \tag{5.5}$$

where $\hat{b}$ ($= \frac{b}{\|b\|}$) is the unit vector in the direction of line b. The $p.\hat{b}$ is the scalar projection of p onto $\hat{b}$ ([69]) and Equation 5.5 is obtained by the triangle formula. Thus, the elbow-point is

$$\text{elbow} = argmax(d) \tag{5.6}$$

The channels before the *elbow* are selected as useful channels (highlighted in green) for classification, and the smaller dataset with this subset of channels is used for the classification step. It is clear from Figure 5.3 that the elbow point can be relaxed, thus allowing a trade-off between data storage and the accuracy of classification.

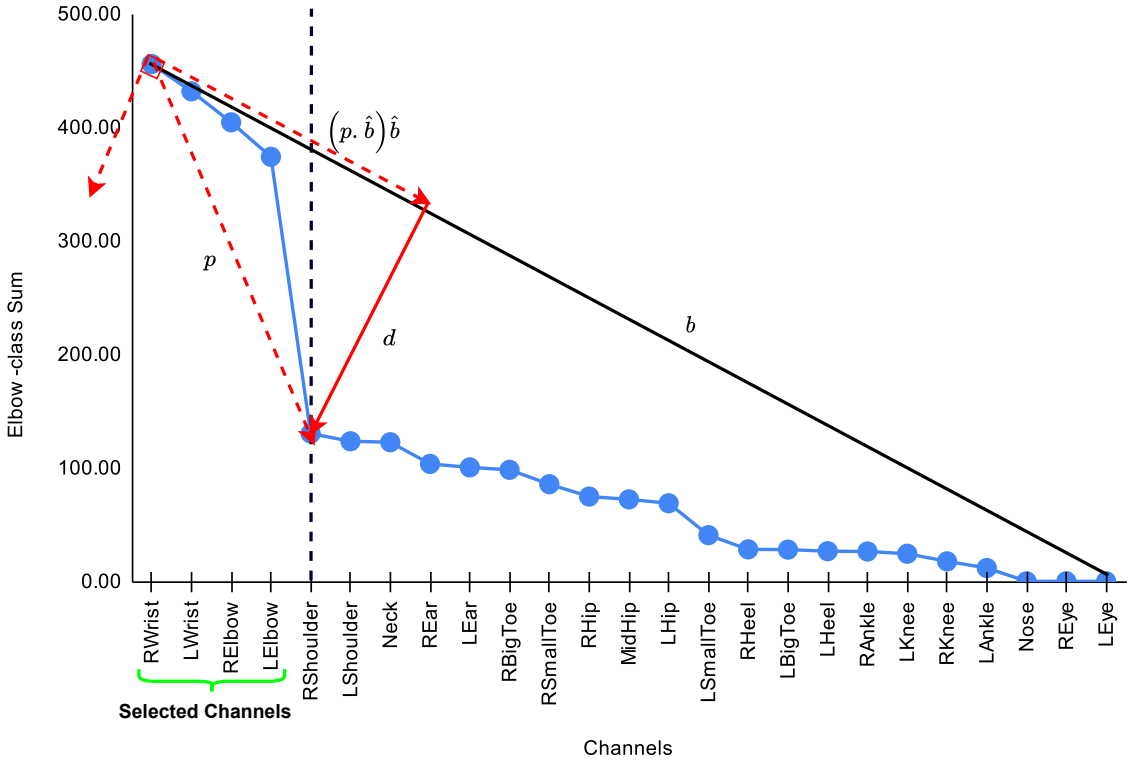


Figure 5.3: An elbow-cut example for channel selection on the Military Press dataset using the *MAD* class prototype. The channels before the elbow point are selected.

5.2.3.3 Channel Ranking

Although the proposed channel selection algorithms can reduce the system's computation requirements, it is not always possible to allocate a clear meaning to selected channels without domain experts. This work presents a method to rank channels according to their discriminative power (where distance between classes is used as a proxy for discrimination). ECS directly works on the sorted distances; therefore, it is easier to rank channels according to the distances. In the case of ECP, the channels selected by ECP are finally ranked based on the sum of distances across classes, similar to the first step in ECS.

5.2.4 Computational Complexity

As mentioned in Section 5.2.1 the dataset is defined with n, d and l as parameters for the number of time series, number of channels, and length of time series. Let c be the number of classes. To analyze the time complexity for our proposed algorithms, we break the techniques into three steps, as illustrated in Figure 5.1, namely: 1. computation of class prototypes, 2. distance matrix formulation, and 3. channel selection.

The time complexity to create the class prototypes is $\sim \mathcal{O}(n \times l \times d)$. The distance matrix stores the distance for each channel for every class pair; therefore, the complexity of computing the distances for each class combination drives the time complexity. Theoretically the time complexity to create a distance matrix with d channels is $\sim \mathcal{O}(c^2 \times l \times d)$. In the final step, the elbow cut takes $\sim \mathcal{O}(d \times log(d))$ to sort and perform elbow cut in ECS and $\sim \mathcal{O}(c^2 \times d \times log(d))$ in ECP.

The total time complexity for channel selection is then the sum of the three steps: compute prototypes + compute class pairwise distances for each channel + sort distances and compute elbow point, which is $\sim \mathcal{O}(n \times l \times d) + \mathcal{O}(c^2 \times l \times d) + \mathcal{O}(c^2 \times d \times log(d))$. In practice, d and c are typically small numbers compared to n and l. It is very rare for a multivariate time series dataset to have more than tens of classes or hundreds of channels. Therefore it is acceptable to assume that $log(d) << l$ and $c^2 < n$. As a result, the final complexity is $\sim \mathcal{O}(n \times l \times d)$. Table 5.10 shows the empirical time and memory results for ECS and ECP and reflects the theoretical complexity derived here.

Regarding space complexity, the only additional information stored is the distance matrix, which grows $\sim \mathcal{O}(c^2 \times d)$. This is insignificant given that the values of d and c are typically small.

5.3 Evaluation on Benchmark Datasets

The approaches designed in this chapter are evaluated using state-of-the-art multivariate time series classification algorithms from the popular Python library *sktime* ([64]). It should be noted that the algorithms are not tuned but are used with the default parameters recommended in the original papers. However, our objective in designing experiments is to understand the relative gain or loss in computational aspects of MTSC algorithms using the proposed channel selection techniques instead of benchmarking MTSC algorithms. All the experiments were run on a Intel(R) Xeon(R) Gold 6140 CPU server with 256 GB RAM.

5.3.1 Datasets

The UEA/UCR Time Series Classification archive [12] is a collection of univariate and multivariate time series data. The repository contains 30 multivariate datasets from various application domains, e.g. ECG, motion classification, and spectra classification. These heterogeneous datasets vary regarding the number of channels (from 2 to 1,345), the number of time series (12 to 30,000) and time series length (8 to 17,894). Here we work with a subset of 26 datasets with equal-length time series. The datasets have predefined splits into train and test sets, and we use these splits for our evaluation. We call this archive the UEA MTSC benchmark.

5.3.2 State-of-the-art MTSC Algorithms

Before analyzing the performance of channel selection, we benchmark existing state-of-the-art algorithms on our system. Table 5.2 shows the performance of classifiers on the 26 datasets. For ease of comparison, we first show the average accuracy over the 26 datasets, as well as the total runtime to complete training and prediction over all datasets. In order to test for the statistical significance of these results, we follow the recommendations in [57, 58, 59]. The accuracy gain is evaluated using a Wilcoxon signed-rank test with Holm correction and visualised with the critical difference (CD) diagram. The CD shows the ranking of methods with respect to their average accuracy rank computed across multiple datasets. Methods that do not have a statistically significant difference in rank, are connected with a thick horizontal line. For computing the CD we use the R library *scmamp*[1] [60]. ROCKET is the best method for scalability and accuracy, closely followed by WEASEL-MUSE and MrSEQL-SAX. However, WEASEL-MUSE and MrSEQL-SAX are slow in comparison to ROCKET. For comparison, in Table 5.3 we show the performance of these classifiers after channel selection with our methods. We also see in Figure 5.4 that there is no significant difference between classifier accuracy before and after channel selection, i.e., with fewer channels and less computation time. We describe next a comparison of our methods to a strong greedy feature selection baseline, as well as a sensitivity analysis for the main components in our methods.

[1] https://github.com/b0rxa/scmamp

Classifiers	Avg. Accuracy	Time (minutes)
ROCKET	71.58	36.93
WEASEL-MUSE	70.28	3774.47
MrSEQL-SAX	66.99	7314.05
Total Memory (data size)	1.2 GB	

Table 5.2: Performance of state-of-the-art multivariate time series classifiers on 26 UEA datasets before channel selection.

Classifiers	Avg. Accuracy	Time (minutes)
ROCKET	73.01	29.26
WEASEL-MUSE	71.26	1104.10
MrSEQL-SAX	68.30	2818.70
Total Memory (data size)	0.38 GB	

Table 5.3: Performance of state-of-the-art multivariate time series classifiers on 26 UEA datasets after channel selection. The best combination of prototype and channel selection approach is used for computing these values. For more details on best performance for each dataset please see Table 5.15.

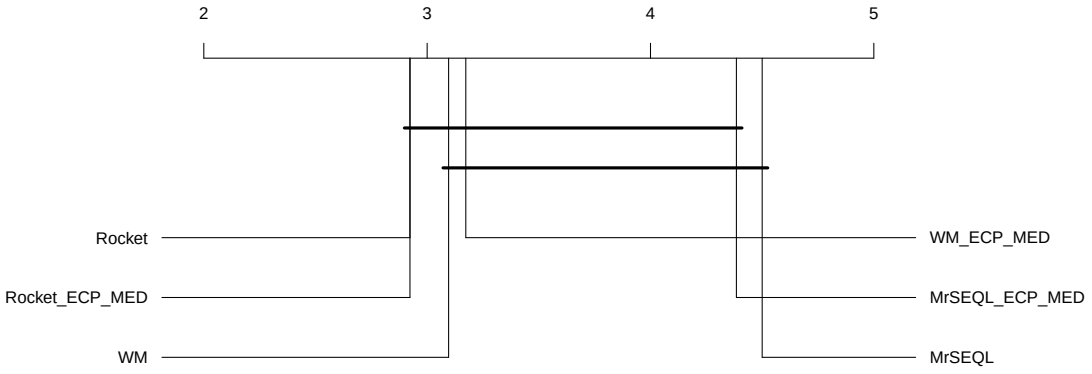


Figure 5.4: Performance of MTSC classifiers with and without channel selection.

5.4 Baseline Comparison: Forward Channel Selection

Filter and Wrapper methods [46] are the methods of choice for performing feature selection in tabular data [70]. The difference between the two methods is that the wrapper methods are exposed to classifier performance while the filter methods are agnostic of any classification evaluation metrics. A wrapper method was proposed in [47] but no code was made available to facilitate comparison, and it is apparent from that work that the method would be computationally expensive. We modify the forward feature selection wrapper method from sckit-learn to a forward channel selection (FCS) method, where a channel of multivariate time series replaces a feature and we use ROCKET as the classifier. There are two types of wrapper methods, namely, forward

selection and backward elimination. This chapter uses tweaked forward selection as a baseline method. We deliberately leave the backward elimination method out as it is slow and requires a lot more memory when compared with forward selection. In default forward selection, the method greedily evaluates channels using a classifier and cross-validation, and provides a ranked list as output. The default setting selects the top 50% channels. As an alternative, domain knowledge can be used to decide the number of channels to be selected. To make the baseline stronger (than the default 50% selection), we tweak this method to select channels starting from the best channel and adding channels until there is no more gain in accuracy (the accuracy tends to increase when starting from the best channel and adding new channels, and then tends to decrease once less informative channels are added). This is similar in nature to using the elbow point to cut the ranked list of channels.

Classifier	Avg. Accuracy	Time (minutes)
ROCKET	71.58	36.93
FCS-ROCKET	70.96	3685.13

Table 5.4: Forward Channel Selection using ROCKET as estimator on 26 datasets.

Table 5.4 shows the performance of the forward channel selection with ROCKET. We find enough evidence of the performance of FCS as it clearly shows that the default classifier, ROCKET, is much faster and more accurate than the baseline, FCS-ROCKET. Therefore it contradicts the primary argument of this work, which is enabling the existing classifier to scale better. It is not feasible to perform forward channel selection for other MTSC methods for computational reasons.

5.4.1 Proposed Channel Selection Methods

In this section, we focus on analysing the impact of different class prototypes and channel selection algorithms on the accuracy and runtime of the MTS classifiers.

5.4.1.1 Comparison of DTW and Euclidean distance

Tables 5.5 and 5.6 compare the performance of DTW and Euclidean distance when creating a distance matrix for ECP class prototypes. While the accuracy of the two distance-based class prototypes is similar, channel selection with Euclidean distance is faster than with DTW. From hereon, we consider Euclidean distance as the default distance metric.

Channel Selection →	ECP		
Class Prototype →	Mean	Median	MAD
Distance function →	EU | DTW	EU | DTW	EU | DTW
Rocket	71.58 | 71.53	71.69 | 71.60	71.86 | 71.58
WeaselMuse	68.71 | 70.30	68.96 | 68.56	68.94 | 68.62
MrSEQL-SAX	67.44 | 66.63	67.35 | 66.83	67.37 | 66.79

Table 5.5: Accuracy for channel selection using Euclidean vs DTW distance for ECP class prototype.

Channel Selection →	ECP		
Class Prototype →	Mean	Median	MAD
Distance function →	EU | DTW	EU | DTW	EU | DTW
Rocket	23.13 | 25	26.40 | 28.24	26.74 | 28.79
WeaselMuse	871.85 | 988.55	843.87 | 1027.60	634.50 | 985.07
MrSEQL-SAX	2067.11 | 2004.19	2117.61 | 2358.81	1178.24 | 2269.91

Table 5.6: Euclidean vs DTW distance computation time (minutes) for ECP class prototype.

Table 5.7 shows the performance of the Median class prototype along with the Mean class prototype. The column ΔAcc shows the difference in accuracy between the default classifier and the one with channel selection. The %Time column shows the amount of time saved due to channel selection. The Median class prototype, and ECP perform better than the Mean class prototype. The ECP-Median channel selection achieves better performance than the default version of ROCKET and MrSEQL-SAX, additionally, saving computation time and memory.

Prototype→	Mean		Median	
Channel Selection→	ECS	ECP	ECS	ECP
Classifier↓	ΔAcc | %Time	ΔAcc | %Time	ΔAcc | %Time	ΔAcc | %Time
ROCKET	-4.41 | 36.74	0.00 | 14.19	-5.02 | 28.51	+0.11 | 18.30
WEASEL-MUSE	-3.80 | 76.90	-1.57 | 67.97	-6.18 | 77.64	-1.32 | 68.02
MrSEQL-SAX	-3.85 | 71.74	+0.45 | 59.65	-3.67 | 71.05	+0.36 | 59.66

Table 5.7: Performance of channel selection on 26 equal-length UEA datasets.

Prototype→	MAD	
Channel Selection→	ECS	ECP
Classifier↓	ΔAcc | %Time	ΔAcc | %Time
ROCKET	-4.53 | 27.59	+0.28 | 17.67
WEASEL-MUSE	-5.77 | 83.19	-1.34 | 74.39
MrSEQL-SAX	-3.31 | 83.39	+0.38 | 77.11

Table 5.8: Performance of channel selection with the MAD prototype on the 26 UEA datasets. ECP-MAD outperforms the other selection strategies.

Although ECP-Median improved the performance, both the class prototypes have their virtues, as described in Section 5.2. The MAD centroid combines both the Mean class prototype and Median class prototype properties. Table 5.8 shows the results when using MAD as the class prototype. The computation time also reduces significantly in the case of WEASEL-MUSE and MrSEQL-SAX.

To further understand the impact of different selection strategies, we have tested another approach for ranking and selecting channels. We first calculate the *l2-norm* of class prototypes for each class instead of the euclidean distance between a class pair. Table 5.9 illustrates the performance of the *l2-norm* on two class-prototypes namely, Mean and Median. Here we examine if channel selection can be performed directly based on the channels' magnitude, rather than the distance between class prototypes. Table 5.9 illustrates the performance of using the *l2-norm* on two-class prototypes namely, Mean and Median.

Prototype→	Mean		Median	
Channel Selection→	ECS	ECP	ECS	ECP
Classifier↓	ΔAcc | %Time	ΔAcc | %Time	ΔAcc | %Time	ΔAcc | %Time
ROCKET	-6.03 | 40.49	-0.21 | 19.36	-4.58 | 30.10	+0.30 | 21.30
WEASEL-MUSE	-4.80 | 80.31	-0.66 | 69.07	-6.07 | 78.06	-2.14 | 72.04
MrSEQL-SAX	-5.14 | 79.60	-0.74 | 71.14	-5.61 | 78.95	-0.04 | 71.49

Table 5.9: Channel selection using the l2-norm instead of distance between class prototypes.

The *l2-norm* performs reasonably when combined with the three classifiers. Although the method enables ECP-Median-l2 with ROCKET to perform similar to the default version ECP-Median with euclidean distance, overall it is not better than the distance-based selection strategies. The reason for this could be that the *l2-norm* of class-prototypes is highly susceptible to data noise for each channel, where a slight jitter could mislead results. Hitherto, from Tables 5.7, 5.8 and 5.9, we conclude that distance-based selection strategies with the ECP-MAD as class prototype works best in comparison to other strategies.

5.4.2 Sensitivity Analysis

The above section shows that channel selection significantly reduces the computation time while maintaining accuracy. In this section, we take a more refined look into the data and analyse the performance of these techniques on the UEA datasets.

5.4.2.1 Channel Utilisation for Different Class Prototypes

Figure 5.5 shows the percentages of channels utilised by the three class prototypes. Interestingly, the higher the number of channels in the dataset, the fewer channels are helpful. Datasets like DuckDuckGeese have 1345 channels; however, only 538 channels are selected by ECP-MAD. As the number of channels decreases, more channels get utilised, which is logical as they tend to possess more information.

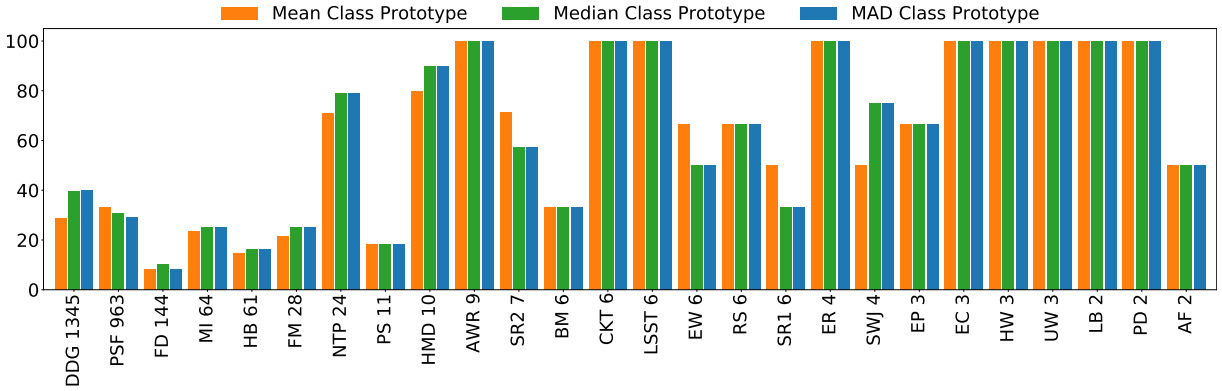


Figure 5.5: Channel utilisation for the three class prototypes with ECP selection. The datasets are arranged in descending order of number of channels.

5.4.2.2 Runtime by Different Class Prototypes

Table 5.10 presents the time and memory required by different class prototypes along with various channel selection techniques. The time taken by MAD is highest among the three, as it iterates over data points for every class (as described in Section 5.2). The table also illustrates the memory required for different datasets. The proposed methods significantly reduce the memory required, as compared to the 1.2GB required to store the original 26 UEA datasets.

Class Prototype→	Mean		Median		MAD	
Channel Selection→	ECS	ECP	ECS	ECP	ECS	ECP
Time (minutes)	0.26	0.27	0.31	0.32	0.64	0.67
Memory (GB)	0.28	0.42	0.27	0.42	0.25	0.41

Table 5.10: Total runtime of channel selection methods and the memory required to store the 26 UEA datasets after selection.

5.4.2.3 Analysis by Datasets

As mentioned in Section 5.3.1, the UEA MTSC archive is a heterogeneous collection of problems. In this section, we discuss the performance of ECP on the benchmark datasets.

Figure 5.6, 5.7 and 5.8 illustrates the change in accuracy on different class prototypes versus the number of channels utilized by each dataset for ROCKET, MrSEQL and WEASEL-MUSE. We note that the accuracy is largely preserved, with a significant reduction in the number of channels, especially for datasets with more than 10 channels. Although the set of channels recommended by the proposed techniques is identical for all classifiers, the performance of classifiers varies for the same set of datasets. The DuckDuckGeese (DDG) dataset gains significant accuracy for ROCKET and MrSEQL but loses some for WEASEL-MUSE. Similarly, for the MotorImagery(MI), MrSEQL-SAX performs better than the default, but ROCKET and WEASEL-MUSE are not as accurate. These results may arise due to the type of classifier, and the study of the classifier is outside the current chapter's scope.

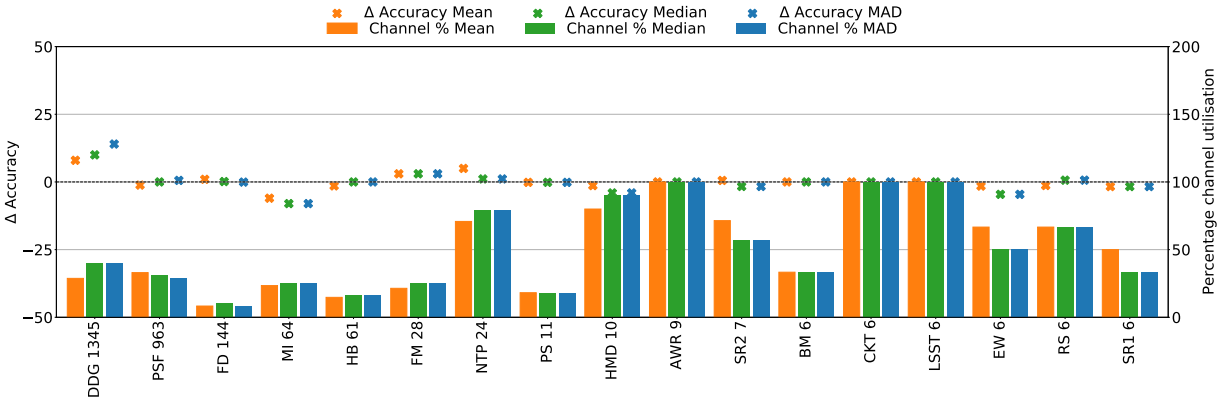


Figure 5.6: ROCKET ECP accuracy and channel utilisation for three class-prototypes.

5.4.2.4 Analysis by the Number of Channels

The UEA datasets vary in the number of channels. The benchmark contains three datasets with more than 100 channels and six datasets with between 10 and 100 channels. Table 5.11 and Table 5.12 demonstrate the performance of channel selection for datasets in these groups. ROCKET outperforms the other two classifiers when the channels are greater than 100. It gains 4.48% average accuracy on the three datasets at the cost of 13.8 seconds (7.04%) more than the default setting.

Although the channel selection for the other two classifiers does not help them as much as in the case of ROCKET, it helps them reduce the time significantly. Moreover,

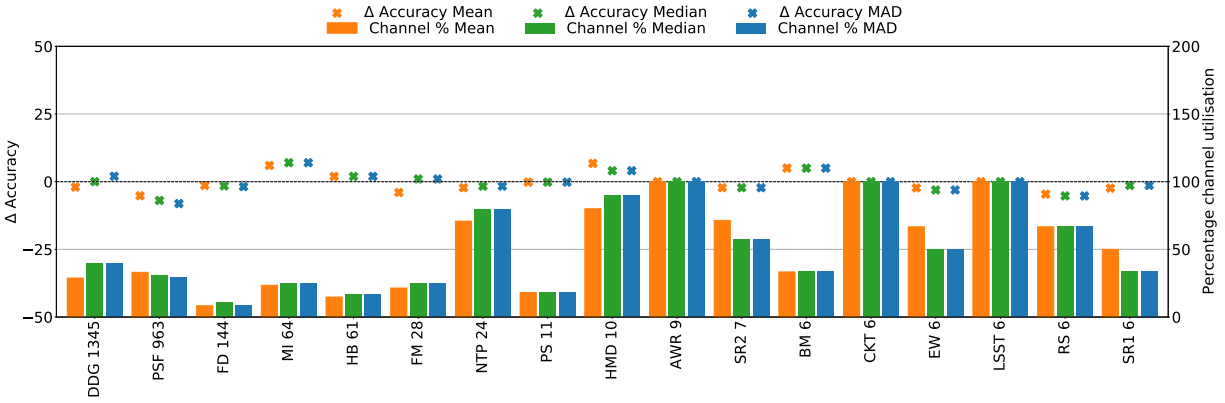


Figure 5.7: MrSEQL-SAX ECP accuracy and channel utilisation for three class-prototypes.

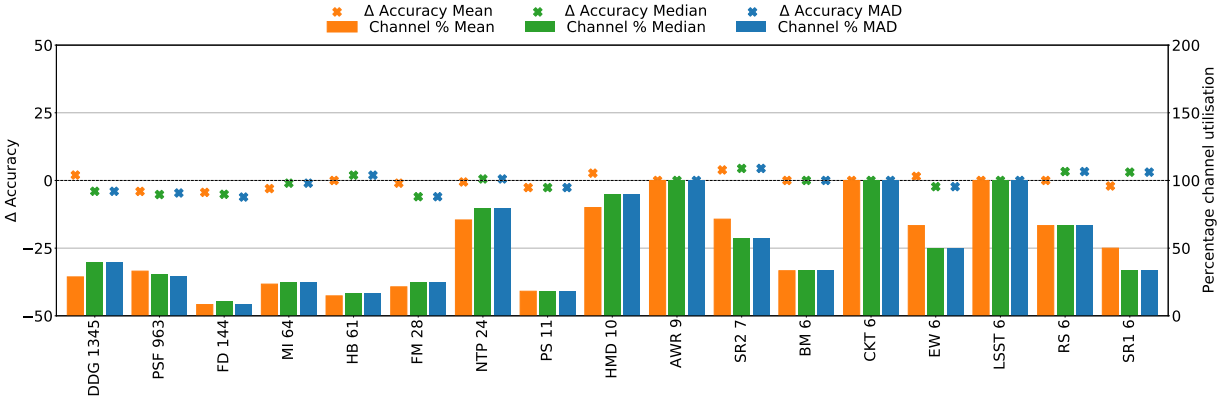


Figure 5.8: WEASEL-MUSE ECP accuracy and channel utilisation for three class-prototypes.

the MrSEQL-SAX performs better than ROCKET and WEASEL-MUSE for datasets between 10 and 100 channels.

Class Prototype→	Mean	Median	MAD
Classifiers↓	ΔAcc | %Time	ΔAcc | %Time	ΔAcc | %Time
ROCKET	+2.60 | 2.30	+3.38 | 1.13	+4.84 | -7.04
WEASEL-MUSE	-2.15 | 76.52	-4.77 | 74.33	-4.92 | 79.06
MrSEQL-SAX	-2.86 | 74.26	-2.83 | 72.62	-2.65 | 84.40

Table 5.11: ECP channel selection on datasets with channels ≥ 100

Class Prototype→	Mean	Median	MAD
Classifiers↓	ΔAcc | %Time	ΔAcc | %Time	ΔAcc | %Time
ROCKET	+0.08 | 20.96	-0.81 | 20.27	-0.81 | 20.47
WEASEL-MUSE	-1.44 | 76.32	-1.42 | 75.03	-1.42 | 84.40
MrSEQL-SAX	+0.31 | 55.91	+1.62 | 55.74	+1.62 | 73.52

Table 5.12: ECP channel selection on datasets with channels > 10 and < 100

5.4.2.5 Analysis by Problem Domains

The application of multivariate data varies across domains. In the UEA archive, the data comes from 5 different domains, namely, Audio Spectra Classification (ASC), EEG/MEG Classification (EEG), Human Activity Recognition (HAR), Motion Classification (MC), ECG Classification (ECG).

Table 5.13 illustrates the performance of the classifiers on the mentioned domains when using the ECP-MAD selection. Channel selection improves the performance of ROCKET in ASR and MC domains, while in the case of EEG, MrSEQL performs slightly better with channel selection than its default setting. Channel selection also aids WEASEL-MUSE in the case of HAR and MC.

Classifiers→	ROCKET | MrSEQL | WM
ASC (3)	+4.48 | +1.26 | -1.55
ECG (2)	-4.00 | -6.67 | -6.66
EEG (6)	-0.28 | +1.10 | -2.29
HAR (9)	-1.20 | -0.55 | +0.18
MC (3)	0.00 | -0.24 | +0.01
Other (3)	+4.44 | -2.69 | -1.54

Table 5.13: Δ Accuracy ECP-MAD channel selection for datasets from different domains.

5.4.2.6 Analysis of Channel Selection on Minority Classes

Although most of the datasets in the UEA benchmark are balanced with regard to the number of samples in each class, we analyse the impact of channel selection on the imbalanced datasets in this section. The significantly imbalanced datasets from the UEA benchmark are reported below with the F1 score (refer to Table 2.1 for class counts). Table 5.14 compares ECP-MAD based channel selection (F1-CS) with the default classifier performance (F1-Default).

Dataset (#Channels)	#Channels Selected	F1-Default	F1-CS	Best-Setting
EW (6)	4	83.36	86.58	ECP+Mean
HW (3)	3	55.72	55.72	ECP+Med
LSST (6)	6	43.66	43.66	ECP+Med
HB (61)	10	63.41	66.87	ECS+MAD

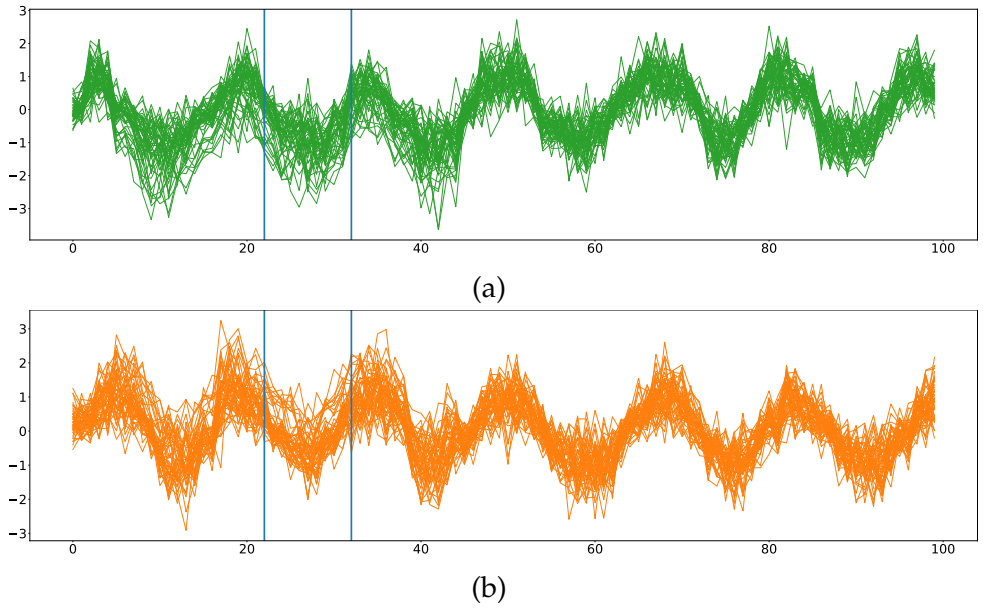
Table 5.14: Analysis of channel-selection on imbalanced datasets.

The datasets EW, HW, LSST, and HB maintain or improve F1 scores with channel selection. For dataset HW and LSST, the channel selection method recommends using

all channels; however, for EW and HB, it is recommended to use fewer channels, especially HB sees a significant reduction in the number of channels to be used. Also, it should be noted that in most cases, ECP works well; however, the class prototypes are crucial for different types of channel selection (refer to Table 5.15).

	Before Channel Selection			After Channel Selection			Best classifier & Accuracy				
Dataset	Rocket	WM	SEQL	CS Rocket	CS WM	CS SEQL	Best Classifier	Best Accuracy	Δ Acc	Best CS	Best CP
DDG 1345	**46.00**	44.00	34.00	**60.00**	50.00	42.00	Rocket	60.00	14.00	ECP	MAD
PSF 963	84.39	**98.27**	95.95	84.97	**94.22**	90.75	WM	94.22	-4.05	ECP	MEAN
FD 144	62.74	**65.21**	55.90	**63.71**	60.81	54.51	Rocket	63.71	0.97	ECP	MED
MI 64	**57.00**	55.00	51.00	51.00	**63.00**	58.00	WM	63.00	8.00	ECS	MEAN
HB 61	**74.15**	73.17	73.17	74.15	**75.12**	75.12	WM	75.12	1.95	ECP	MED
FM 28	53.00	54.00	**56.00**	56.00	**60.00**	57.00	WM	60.00	6.00	ECS	MEAN
NTP 24	87.22	**91.67**	87.22	**92.22**	92.22	85.56	Rocket	92.22	5.00	ECP	MED
PS 11	27.62	**31.46**	26.27	27.47	**29.29**	26.10	WM	29.29	-2.17	ECS	MED
HMD 10	**54.05**	28.38	14.86	**54.05**	31.08	22.97	Rocket	54.05	0.00	ECP	MED
AWR 9	**99.33**	99.33	99.33	**99.33**	99.33	99.33	Rocket	99.33	0.00	ECP	MED
SR2 7	**53.89**	47.78	50.56	55.00	**56.11**	48.33	WM	56.11	8.33	ECS	MEAN
BM 6	**100.00**	100.00	95.00	**100.00**	100.00	100.00	Rocket	100.00	0.00	ECP	MED
CKT 6	**100.00**	100.00	98.61	**100.00**	100.00	98.61	Rocket	100.00	0.00	ECP	MED
EW 6	54.62	**89.31**	73.28	54.62	**90.84**	70.99	WM	90.84	1.53	ECP	MEAN
LSST 6	**89.31**	61.27	58.80	**87.79**	61.27	58.80	Rocket	87.79	-1.52	ECP	MED
RS 6	**90.79**	86.18	86.84	**91.45**	89.47	82.24	Rocket	91.45	0.66	ECP	MED
SR1 6	**84.64**	78.50	68.26	**82.94**	81.57	66.89	Rocket	82.94	-1.70	ECP	MED
ER 4	**98.15**	96.67	87.78	**98.15**	96.67	87.78	Rocket	98.15	0.00	ECP	MED
SWJ 4	**53.33**	46.67	33.33	53.33	**60.00**	40.00	WM	60.00	13.33	ECP	MED
EP 3	98.55	**100.00**	99.28	**100.00**	100.00	98.55	Rocket	100.00	1.45	ECP	MED
EC 3	43.35	36.50	**55.51**	49.81	44.49	**58.17**	SEQL	58.17	2.66	ECP	MAD
HW 3	**59.53**	26.12	47.41	**59.53**	26.12	47.41	Rocket	59.53	0.00	ECP	MED
UW 3	**94.06**	90.63	87.19	**94.06**	90.63	87.19	Rocket	94.06	0.00	ECP	MED
AF 2	6.67	**40.00**	26.67	20.00	13.33	**40.00**	SEQL	40.00	13.33	ECS	MED
LB 2	90.56	**93.33**	87.22	90.56	**93.33**	87.22	WM	93.33	0.00	ECP	MED
PD 2	**98.20**	93.85	92.28	**98.20**	93.85	92.28	Rocket	98.20	0.00	ECP	MED

Table 5.15: Accuracy performance of channel selection on UEA MTSC datasets with different channel selection and channel prototypes. Δ Acc represents the difference in accuracy achieved by a classifier with and without channel selection.



(a)

(b)

Figure 5.9: Two synthetic (multivariate) time series of different classes generated by the Pseudo Periodic process. The important time steps are 22-32 where the values in (a) are typically lower than (b).

5.5 Case Study 1: Synthetic Datasets

The UEA benchmark suggests that our channel selection methods can reduce the computation time and memory while at least maintaining the same level of classification accuracy. However, other than using the resulting accuracy, we were unable to ascertain the effectiveness of the channel selection (i.e., did it select the right channels?) because we have no such ground truth.

To address this problem, we make use of synthetic datasets from the work of [3], where the important channels and timepoints are set during the data generation process. We select the RareTime setting, where a small part of the time series is informative. The dataset type is generated using three signal types: Gaussian Process (GP), Pseudo Periodic (PP), and AutoRegressive (AR). The position of the signal can be stationary or moving (can change position within the channel). The length of time series is fixed to 100, and length of the important time series segment is 10. There are two classes.

The number of important channels is fixed to 40. Therefore, as the number of channels increases, the ratio of the important channel to all channels decreases. For example, from 40%, it goes to 20% and then 10% for 100, 200 and 400 channels, so the problem becomes more difficult with an increasing number of channels. Figure 5.10 presents the performance of ROCKET on the synthetic data. ROCKET is the most accurate classifier among the three we studied; therefore, we use ROCKET to analyse the synthetic data.

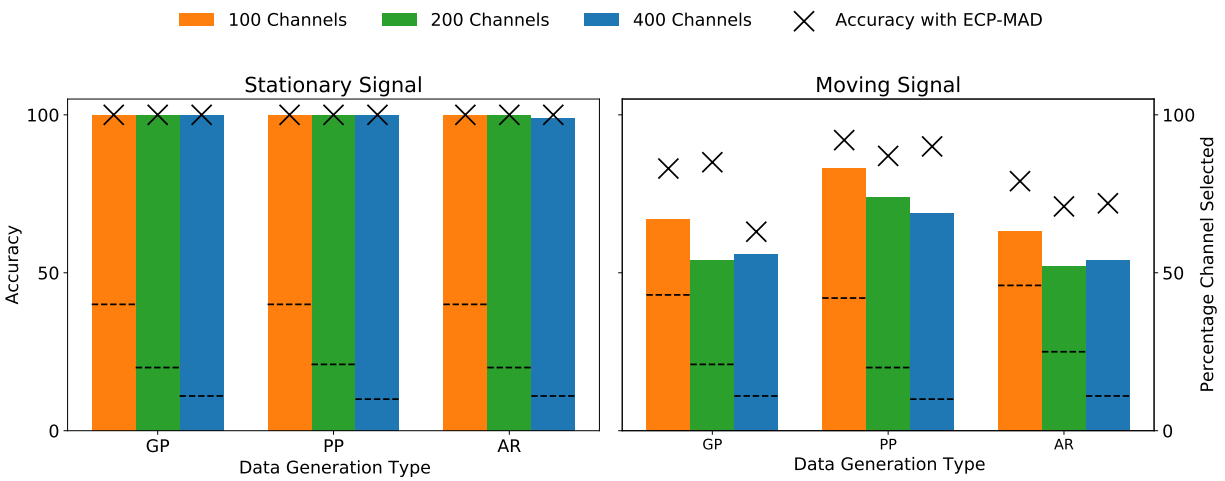


Figure 5.10: Synthetic datasets generated using the procedure in [3] using the RareTime setting. The bar height shows the accuracy before channel selection, while the cross sign shows the accuracy after channel selection. The dashed black line indicates the percentage of channels selected.

ROCKET achieves 100 percent accuracy for the stationary signal; however, it struggles to perform accurately when the signal is moving. When we use ROCKET with our channels selection method, ECP-MAD, then ROCKET achieves the same accuracy as without channel selection in the case of the stationary signal data; furthermore, in the case of moving signal, channel selection enables ROCKET to perform significantly better.

The interesting point is that channel selection extracts approximately the same number of important channels as used during dataset generation. For 40% of important channels in 100 channel stationary signal data, the channel selection recommends 40% channels in static signal data for all three signals, and 43%, 42% and 46% for moving signal for GP, PP, and AR signal data, respectively. Similarly, for 200 channel data, the important channels created are 40, the number of channels recommended is 20%, 21% and 20% for stationary signal data and 21, 20 and 25 for moving signal data in GP, PP and AR, respectively. With the 400 channels, the number of important channels in the dataset reduces to 10%, and it becomes harder for the classifier to find useful signal; however, the channel selection algorithm reduces the efforts required by the classifier. The method selects 11%, 10% and 11% for stationary GP, PP and AR signal data and 14%, 11% and 13% for moving GP, PP and AR signal data, respectively.

5.5.1 Performance of Channel Selection on Minority Classes

In continuation to Section 5.4.2.6, we analyse the impact of channel selection on minority classes in synthetic data using ROCKET. Table 5.16 presents the F1 score for moving

and stationary signals in GP, PP and AR datasets with 100 channels across the different scenarios of class imbalance. We vary the class ratio to evaluate the performance of channel selection. This enables us to study how the channel selection performs with different cardinalities.

Datasets	Class Ratio	F1-Moving	F1-Stationary
	#0 : #1	Default | CS	Default | CS
GP	50:50	83.00 | 82.98	100 | 100
	90:10	34.21 | 34.21	81.00 | 97.99
	80:20	34.21 | 40.78	93.93 | 100
PP	50:50	92.00 | 92.00	100 | 100
	90:10	33.33 | 35.51	100 | 100
	80:20	35.52 | 61.55	100 | 100
AR	50:50	79.00 | 78.64	100 | 100
	90:10	29.07 | 29.07	77.52 | 94.97
	80:20	29.07 | 34.54	97.99 | 98.99

Table 5.16: Analysis of ECP-MAD channel selection in minority class on synthetic data.

Similar to the analysis above, we find that the cardinality of a class does not impact the channel selection. However, with various levels of imbalance, even a good classifier like ROCKET struggles to perform better. The channel selection, on some occasions, improves accuracy; however, not as much as in the case of balanced data.

5.6 Case Study 2: Military Press Dataset

The experiments in the above sections confirm that our channel selection methods enable the state-of-the-art classifiers to achieve better accuracy in significantly smaller amounts of time and memory and, in the worst case, saves time and memory with minimal loss of accuracy. However, to perform a sanity check on real-world data, we provide an additional study on Military Press data, collected and verified at University College Dublin with the help of domain experts.

5.6.1 Dataset

A total of 56 healthy volunteers (34 males and 22 females; age: 26 ± 5 years, height: 1.73 $\pm$ 0.09 m and body mass: 72 ± 15 kg) participated in a study aimed at analysing the execution of the Military Press strength and conditioning exercise. The participants completed ten repetitions of the normal form and ten repetitions of induced forms. The NSCA guidelines were applied under the guidance of sports physiotherapists and

conditioning coaches to ensure standardisation. The dataset was extracted from the video of individuals performing the exercise with the help of the human body pose estimation library OpenPose[2]. There are four classes in the dataset, namely: Normal (N), Asymmetrical (A), Reduced Range (R) and Arch (arch). The N refers to the correct execution of the exercise; A refers to when the barbell is lopsided and asymmetrical, R refers to the form where the bar is not brought down completely to the shoulder level and Arch refers to when participants arch their back. A total of 25 body parts were tracked along X and Y axis, as seen in Figure 5.3. These 50 body trackers (25×2) act as channels for the MTSC task. The train and test size for this dataset is 1452 and 601 respectively and the length of time-series is 160.

5.6.2 Evaluation

When used with all 50 channels, ROCKET takes 1.45 minutes and achieves an accuracy of 77.53%. In this setting, the normalisation hyperparameter of ROCKET is switched off; this is because the signal magnitude contains crucial information. Table 5.17 shows accuracy and computation results before and after channel selection.

	Before Selection	After Selection	
Classifiers ↓	Acc | Time | Mem (MB)	Acc | Time | Mem (MB)	Best Setting
ROCKET	77.53 | 1.45 | 98.5	84.02 | 1.52 | 13.80	ECS-Median
WEASEL-MUSE	57.57 | 71.55 | 98.5	59.90 | 38.87 | 43.38	ECP-Median
MrSEQL-SAX	61.56 | 633.56 | 98.5	59.07 | 360.80 | 43.38	ECP-Median

Table 5.17: The SOTA classifiers on 50 Channel Military Press dataset.

5.6.3 Performance of SOTA Classifiers on the Military Press Dataset

Table 5.18 demonstrates the performance of channel selection on the Military Press dataset. The ECS with Median as class prototype outperforms any other setting; the gain in accuracy is about 7 percentage points. All the other channel selection methods perform better than ROCKET in its default setting.

Ranking: The channel ranking using the best method here, ECS with the median prototype, is as follows: 'RWrist_Y', 'LWrist_Y', 'RElbow_Y', 'LElbow_Y', 'RElbow_X', 'LElbow_X', 'RHeel_Y'. We note that out of 50 channels, this method selects only a subset of 7 channels, which not only reduces the data drastically, but also results in significantly

[2] https://github.com/CMU-Perceptual-Computing-Lab/openpose

Center→	Median		MAD	
Channel Selection→	ECS	ECP	ECS	ECP
Classifier↓	Acc | Time	Acc | Time	Acc | Time	Acc | Time
ROCKET	84.02 | 1.52	81.03 | 1.78	82.59 | 1.59	80.20 | 1.75
WEASEL-MUSE	54.57 | 10.28	59.90 | 38.87	54.57 | 10.23	58.57 | 48.35
MrSEQL-SAX	58.90 | 58.63	59.07 | 360.80	58.90 | 58.63	51.32 | 382.08

Table 5.18: Performance of SOTA classifiers on MP dataset.

increased accuracy. We consider this a very good result, also considering that it makes it easier to provide feedback to the participant executing the exercise, by pointing to the most important body parts for the execution.

5.6.4 Performance of Channel Selection with Two Classes

With the help of domain experts, we know that the Military Press exercises with classes Asymmetric, Reduced, and Arch, are exercises done incorrectly. All three classes have different signal magnitudes. To study the impact of channel selection when one of the classes has a group of different behaviours, we group the classes mentioned above into Abnormal and train a binary classifier, Abnormal, against the Normal class in the dataset. Since there are only two classes, ECS and ECP strategies are identical.

All	Mean	Median	MAD
72.05	73.71	77.05	77.50

Table 5.19: F1 score for ROCKET when using two classes on the MP dataset

Table 5.19 illustrates the performance of ROCKET on the three prototypes used. We see that channel selection helps ROCKET to distinguish between the two classes. Furthermore, MAD is the most effective strategy. Since this is an imbalanced dataset, we also compared precision and recall in Table 5.20 and found that, with channel selection, there is an improvement in both precision and recall.

	Precision	Recall	F1
Abnormal	0.86	0.86	0.86
Normal	0.59	0.57	0.58

F1 results with all channels.

Precision	Recall	F1	Support
0.90	0.87	0.88	451
0.64	0.69	0.67	150

F1 results for the MAD prototype.

Table 5.20: Comparison of performance of ROCKET with and without channel selection.

5.7 Limitations & Future Work

While ECP performs better than ECS in most datasets, it may collect some noisy channels when iterating through the different class pairs in the distance matrix. These noisy channels, along with useful channels, are then used for classification and this could negatively impact the classifier accuracy. The proposed methods rely on the euclidean distance between two class prototypes; however, if a class prototype is too noisy, it could amplify the euclidean distance. With the help of the signal's magnitude, the euclidean distance could be regularised. In our future work we want to further analyse the impact of combining information from the prototype magnitude and the distance-based computation to address this issue.

We also plan to investigate whether combining multiple class prototypes can further improve results. At the moment we only use one class prototype to represent each class and for classes with subgroups of behaviours this could be an issue. We can consider first clustering the behaviours in each class and representing each subgroup with a prototype. In that case, we also need to investigate whether all class prototypes are equally important, or if some are more important than others.

The current study focuses on channel selection based on euclidean distance and does not use time-series attributes or statistical properties of each class to adjust the selection strategy. Similar to channel selection, we could also explore the time series segments that contribute most to the classification, providing another avenue to make existing MTSC algorithms more scalable.

5.8 Conclusion

This study shows that for many MTSC datasets, state-of-the-art classifiers do not require all the channels to achieve the best accuracy. Some channels are very noisy for the classification task and act as a bottleneck for the classifier to achieve its maximum potential. We observed that our channel selection methods remove data noise and drastically reduce the required computation time for existing MTSC methods. The current study showed that the distance between the class prototypes of various channels plays a crucial role in identifying the noisy channels. Our channel selection strategies, ECP and ECS, can select the useful channels based on euclidean distance between class prototypes. Our techniques significantly reduced the runtime and memory required to run MTSC classifiers. There is some accuracy loss for some datasets; however, this

is minimal, and the benefit of data reduction and computation savings outweigh the small accuracy loss.

Channel selection saves approximately 75% time and around 60% memory. The proposed channel selection techniques have more impact on the datasets with a higher number of channels. We verified the performance of our channel selection methods on various types of signals with varying levels of difficulty, ranging from the popular UEA MTSC benchmark to synthetic data and a real-world case study. The channels selected performed very well on those datasets, enhancing the performance of the most accurate classifier, ROCKET. On our case study with a real-world dataset, Military Press, our channel selection methods combined with ROCKET improved the accuracy by more than 8%. The proposed channel selection techniques are classifier-agnostic, they can be plugged into any classification pipeline, before training an MTSC classifier. They are also simple, fast and at the same time robust to different types of noise, which leads to preserved classifier accuracy as shown in our extensive experiments.

In the next chapter, we compare some of the recent SOTA MTSC classifiers with classical tabular classifiers. This gives us an opportunity to look at the recent development in MTSC literature as well as perform a sanity on modern time series classifiers.

CHAPTER

SIX

BENCHMARKING THE SCALABILITY-ACCURACY TRADE-OFFS FOR TABULAR VERSUS TIME SERIES CLASSIFICATION METHODS

6.1 Introduction

While the previous chapters delved into the scalability of multivariate time series classifiers, recent advancements in the time series classification domain have emerged. The ever-evolving computational capabilities and abundant applications and use cases have led to the development of a wide range of time series classification methods, from simple distance-based methods (1-NN-DTW [14] to complex deep learning models (Inception Time [44]) and more recently the ROCKET family [31, 20, 71].

Notably, these methods have not been compared with traditional machine learning approaches such as Linear Discriminant Analysis [72], Logistic Regression [73], Ridge Regression [74], and Random Forest [75], leaving a gap in understanding their relative performance. Most of the research in time series classification is focused on establishing state-of-the-art results, developing scalable algorithms, and making models explainable. However, in this quest, it is often possible to forget the first principle of research, which is to compare with existing simpler methods.

Historically, there have been many instances where traditional models have outperformed deep learning methods on some tasks. For example, a recent study [76] showed that linear models can be more effective than deep learning networks for forecasting. Similarly, the work of [77, 78] showed that linear models can outperform other complex models for classification tasks in spectroscopy data. However, there is less empirical

work investigating the performance of classic tabular models on time series classification tasks.

In this chapter, we take a step back from the pursuit of providing yet another state-of-the-art method and perform some simple sanity checks, which are often missed. Unlike time-series models that explicitly account for temporal patterns, tabular models ignore such dependencies. We compare the performance of tabular models with the ROCKET [31, 20, 71] family of classifiers, which are currently considered state-of-the-art for time series classification.

In this chapter, the main contributions are:

- We empirically compare tabular and time series methods on the established UCR/UEA benchmarks for univariate and multivariate time series classification.
- We discuss the performance of tabular versus time series methods for different data and problem types and the potential implications for how the very popular UCR/UEA benchmarks are formed and used by the community. In particular, if tabular methods significantly outperform time series methods on some problem types, we raise the question of whether these datasets should be included in a time series benchmark.

6.2 Experiments

6.2.1 Datasets & Evaluation

Datasets: The UEA/UCR [79] benchmark datasets are mostly used in the empirical evaluation and comparison of various algorithms. Since the benchmark contains both univariate and multivariate datasets, it is popular for testing new algorithms on Table 2.1 and 6.1 provides the data dictionary for both types of datasets. As is common in recent time series literature, we run experiments on 109 univariate datasets and 25 equal-length multivariate datasets; we exclude Pen Digits as Minirocket requires a minimum time series length of 8 to run successfully.

Evaluation: This chapter delves into a comparative analysis of the performance of tabular and time-series methods. The primary objective is to highlight the performance discrepancy between these two approaches by employing percentage point differences. Additionally, we perform domain-specific and accuracy-time trade-off analysis.

In Figure 6.4 and Figure 6.9, we illustrate the accuracy of tabular and time series models on each dataset, focusing on comparing the best-performing tabular with the best-

performing time series model. The plot is divided into three distinct regions: green, grey, and red.

- The green region illustrates the datasets where the tabular models outperform the time series models or where both models achieve the same accuracy.
- The grey region represents datasets where the two models have performance within a fixed threshold. It is crucial to consider the accuracy-time trade-off in this region when deciding the better model. Datasets in this region are highlighted when the difference between the best-performing time-series model and the best-performing tabular model ranges from 1 to 9 percentage points.
- The red region represents the datasets where time series models outperform tabular models. The time series models in these datasets are at least ten percentage points better than tabular models.

Table 6.1: Data dictionary for Univariate time series classification.

Data	Train Size	Test Size	TS-Len	#Classes	Domain
ACSF1	100	100	1460	10	DEVICE
Adiac	390	391	176	37	IMAGE
ArrowHead	36	175	251	3	IMAGE
Beef	30	30	470	5	SPECTRO
BeetleFly	20	20	512	2	IMAGE
BirdChicken	20	20	512	2	IMAGE
BME	30	150	128	3	SIMULATED
Car	60	60	577	4	SENSOR
CBF	30	900	128	3	SIMULATED
Chinatown	20	345	24	2	Traffic
ChlorineConcentration	467	3840	166	3	SIMULATED
CinCECGTorso	40	1380	1639	4	ECG
Coffee	28	28	286	2	SPECTRO
Computers	250	250	720	2	DEVICE
CricketX	390	390	300	12	MOTION
CricketY	390	390	300	12	MOTION
CricketZ	390	390	300	12	MOTION
Crop	7200	16800	46	24	IMAGE
DiatomSizeReduction	16	306	345	4	IMAGE
DistalPhalanxOutlineAgeGroup	400	139	80	3	IMAGE
DistalPhalanxOutlineCorrect	600	276	80	2	IMAGE
DistalPhalanxTW	400	139	80	6	IMAGE
Earthquakes	322	139	512	2	SENSOR
ECG200	100	100	96	2	ECG
ECG5000	500	4500	140	5	ECG
ECGFiveDays	23	861	136	2	ECG
ElectricDevices	8926	7711	96	7	DEVICE
EOGHorizontalSignal	362	362	1250	12	EOG
EOGVerticalSignal	362	362	1250	12	EOG
EthanolLevel	504	500	1751	4	SPECTRO
FaceAll	560	1690	131	14	IMAGE
FaceFour	24	88	350	4	IMAGE
FacesUCR	200	2050	131	14	IMAGE

FiftyWords	450	455	270	50	IMAGE
Fish	175	175	463	7	IMAGE
FordA	3601	1320	500	2	SENSOR
FordB	3636	810	500	2	SENSOR
FreezerRegularTrain	150	2850	301	2	SENSOR
FreezerSmallTrain	28	2850	301	2	SENSOR
GunPoint	50	150	150	2	MOTION
GunPointAgeSpan	135	316	150	2	MOTION
GunPointMaleVersusFemale	135	316	150	2	MOTION
GunPointOldVersusYoung	135	316	150	2	MOTION
Ham	109	105	431	2	SPECTRO
Haptics	155	308	1092	5	MOTION
Herring	64	64	512	2	IMAGE
HouseTwenty	34	101	3000	2	DEVICE
InlineSkate	100	550	1882	7	MOTION
InsectEPGRegularTrain	62	249	601	3	EPG
InsectEPGSmallTrain	17	249	601	3	EPG
ItalyPowerDemand	67	1029	24	2	SENSOR
LargeKitchenAppliances	375	375	720	3	DEVICE
Lightning2	60	61	637	2	SENSOR
Lightning7	70	73	319	7	SENSOR
Mallat	55	2345	1024	8	SIMULATED
Meat	60	60	448	3	SPECTRO
MedicalImages	381	760	99	10	IMAGE
MiddlePhalanxOutlineAgeGroup	400	154	80	3	IMAGE
MiddlePhalanxOutlineCorrect	600	291	80	2	IMAGE
MiddlePhalanxTW	399	154	80	6	IMAGE
MixedShapes	500	2425	1024	5	IMAGE
MixedShapesSmallTrain	100	2425	1024	5	IMAGE
MoteStrain	20	1252	84	2	SENSOR
OliveOil	30	30	570	4	SPECTRO
OSULeaf	200	242	427	6	IMAGE
PhalangesOutlinesCorrect	1800	858	80	2	IMAGE
Phoneme	214	1896	1024	39	SOUND
PigAirwayPressure	104	208	2000	52	HEMODYNAMICS
PigArtPressure	104	208	2000	52	HEMODYNAMICS
PigCVP	104	208	2000	52	HEMODYNAMICS
Plane	105	105	144	7	SENSOR

PowerCons	180	180	144	2	DEVICE
ProximalPhalanxOutlineAgeGroup	400	205	80	3	IMAGE
ProximalPhalanxOutlineCorrect	600	291	80	2	IMAGE
ProximalPhalanxTW	400	205	80	6	IMAGE
RefrigerationDevices	375	375	720	3	DEVICE
Rock	20	50	2844	4	SPECTRO
ScreenType	375	375	720	3	DEVICE
SemgHandGenderCh2	300	600	1500	2	SPECTRO
SemgHandMovementCh2	450	450	1500	6	SPECTRO
SemgHandSubjectCh2	450	450	1500	5	SPECTRO
ShapeletSim	20	180	500	2	SIMULATED
ShapesAll	600	600	512	60	IMAGE
SmallKitchenAppliances	375	375	720	3	DEVICE
SmoothSubspace	150	150	15	3	SIMULATED
SonyAIBORobotSurface1	20	601	70	2	SENSOR
SonyAIBORobotSurface2	27	953	65	2	SENSOR
StarLightCurves	1000	8236	1024	3	SENSOR
Strawberry	613	370	235	2	SPECTRO
SwedishLeaf	500	625	128	15	IMAGE
Symbols	25	995	398	6	IMAGE
SyntheticControl	300	300	60	6	SIMULATED
ToeSegmentation1	40	228	277	2	MOTION
ToeSegmentation2	36	130	343	2	MOTION
Trace	100	100	275	4	SENSOR
TwoLeadECG	23	1139	82	2	ECG
TwoPatterns	1000	4000	128	4	SIMULATED
UMD	36	144	150	3	SIMULATED
UWaveGestureLibraryAll	896	3582	945	8	MOTION
UWaveGestureLibraryX	896	3582	315	8	MOTION
UWaveGestureLibraryY	896	3582	315	8	MOTION
UWaveGestureLibraryZ	896	3582	315	8	MOTION
Wafer	1000	6164	152	2	SENSOR
Wine	57	54	234	2	SPECTRO

6.2.2 Multivariate Time Series Classification

Before comparing tabular versus time-series models, we compared a few popular methods within each group separately.

The UEA/UCR MTSC benchmark dataset we utilized for this analysis comprised 26 datasets. However, to ensure consistency and comparability among the models, we narrowed down our focus to the 25 datasets that all models could run on. We filtered the datasets based on equal length, and one dataset (Pen Digits) was removed due to Minirocket, which cannot run on datasets with lengths less than 8.

Data Preprocessing: Unlike univariate time series, which have data from a single channel, multivariate time series data have multiple channels. To convert this data into a format that a tabular model can process, we first standardize each channel's data and then concatenate the data across all channels.

Tabular Methods Results. For tabular methods, we select three linear methods, namely, Linear Discriminant Analysis, Logistic Regression and Ridge RegressionCV, known for their efficiency and effectiveness in real-world applications [77], as well as Random Forest to have an effective non-linear classifier. We run these methods using the sklearn implementation[1] with default parameters. Later in the paper, we also discuss parameter tuning and its impact on accuracy and runtime. The critical difference diagram (Figure 6.1) illustrates that Random Forest performed significantly better than the other three models, and Logistic Regression outperformed the other two linear models.

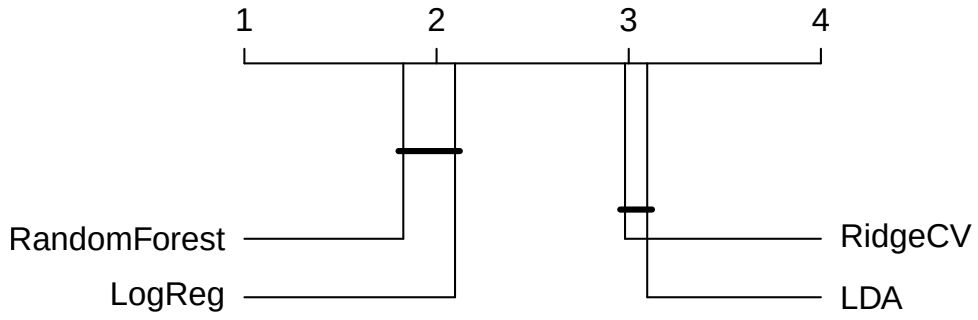


Figure 6.1: Accuracy comparison of tabular methods on MTSC datasets.

Table 6.2 shows the total time taken by tabular models and their corresponding mean accuracy. The table corroborates the results of the critical difference diagram, which showed that Random Forest is the most accurate tabular model, closely followed by LogisticRegression and RidgeCV. RidgeCV is also the most time-efficient method.

[1] https://scikit-learn.org/stable/supervised_learning.html

Table 6.2: Mean accuracy and total computation time taken by tabular models on MTSC datasets.

	Mean Accuracy	Total Time (minutes)
RandomForest	0.61	6.40
LogisticRegression	0.59	6.20
RidgeCV	0.56	5.27
LDA	0.52	6.70

Time Series Methods Results. Similar to the tabular methods, we ran the multivariate time series methods, namely Minirocket, Multirocket, and Rocket, on the MTSC datasets. Since the implemented algorithm works well with multivariate time series, there was no need to preprocess the data in this case.

Figure 6.2 and Table 6.3 illustrate the performance of time series methods on the benchmark datasets. Both the figure and table show that Minirocket outperforms the other two classifiers. Additionally, Minirocket is also the fastest method among the three methods.

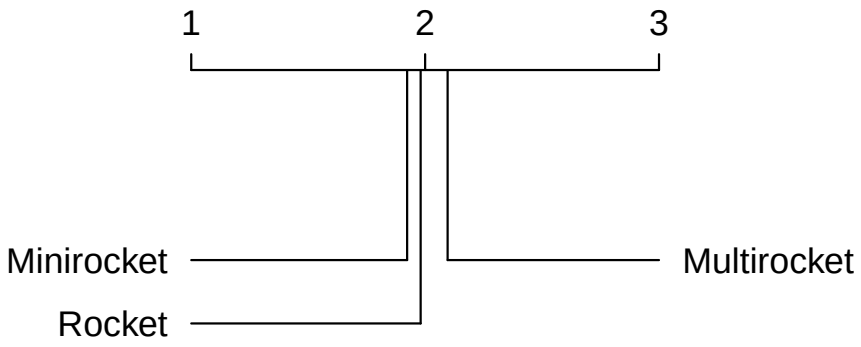


Figure 6.2: Accuracy comparison of time-series methods on MTSC datasets.

Table 6.3: Mean accuracy and total computation time taken by time series models on MTSC datasets.

	Mean Accuracy	Total Time (minutes)
Minirocket	0.71	49.33
Multirocket	0.70	67.10
Rocket	0.70	129.05

6.2.2.1 Time Series Methods vs Tabular Methods.

Finally, we compared tabular and time series models, as shown in Figure 6.3. As expected, the time series models outperformed the tabular models in terms of average

accuracy. However, we conducted a more detailed analysis to investigate the reasons for this difference. We discuss our findings below.

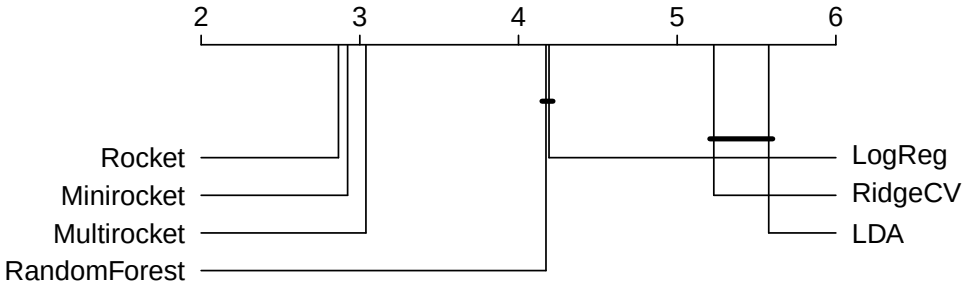


Figure 6.3: Accuracy comparison time-series and tabular methods on MTSC datasets.

Figure 6.4 shows the difference in performance between the best-performing tabular model and the best-performing time series model. The performance of each model is highlighted in a different region, as defined above in Section 6.1. Approximately 28 percent of the datasets are represented in each green and grey region (56 percent total), indicating that the tabular model performs better or within 10 percentage points in these cases. Another 44 percent of the datasets fall within the red region, indicating that the time series models outperform the tabular models in those instances.

6.2.2.2 Computation Time Analysis

For the same reasons as for the univariate time series classification task, we perform the time-accuracy tradeoff analysis for multivariate time series classification. Figure 6.5 illustrates the performance of various time-series and tabular models on the datasets in the grey region of Figure 6.4. Rocket is the most accurate among time-series models, and Random Forest is the most accurate model among tabular models. The difference between the mean accuracy of Rocket and the mean accuracy of Random Forest is about 5 percentage point, while the difference in total computation time is about 4 minutes.

6.2.2.3 Domain Wise Analysis

In addition to considering the trade-off between time and accuracy, we also analyzed the domain-wise performance of tabular and time series models in multivariate datasets in Table 6.4. The datasets consisted of 6 domains, with 60% of the data coming from two domains (HAR and EEG). Time series models generally performed well, but tabular models performed better in the ECG and EEG/MEG domains.

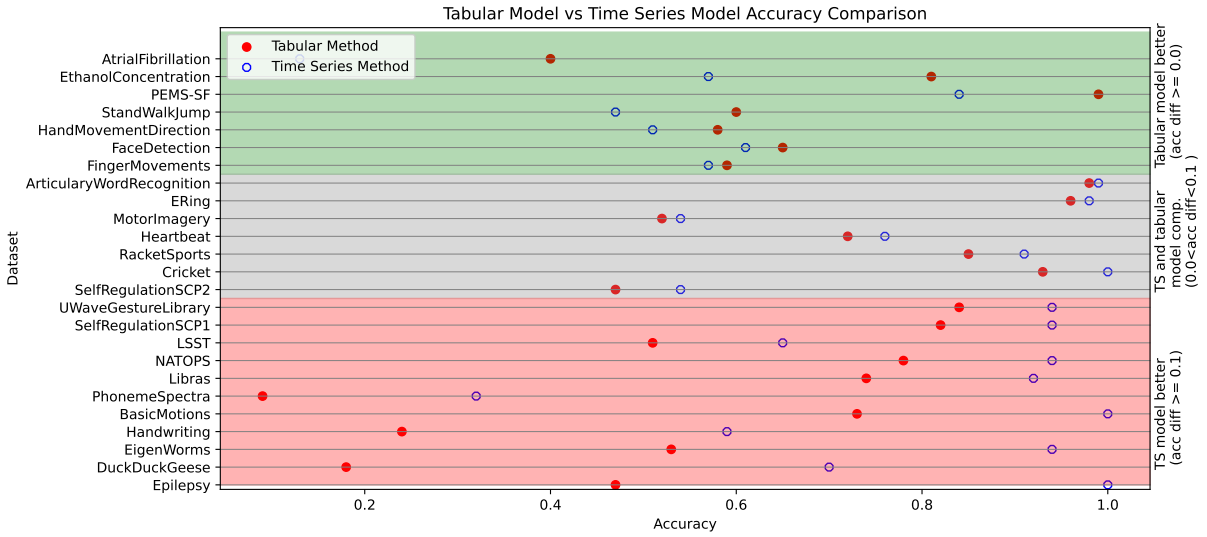


Figure 6.4: Accuracy comparison of time series models with tabular models on multivariate time series datasets. Red circles represent tabular models, and blue circles represent time series models. Each marker shows the maximum accuracy achieved by the tabular models versus the time series models. Detailed results are provided with the code.

Table 6.4: Mean accuracy of classifiers by problem types on UCR multivariate datasets.

	Tabular Models				Time Series Models		
Domain (#datasets)	RidgeCV	LDA	LogReg	RandomForest	Rocket	Minirocket	Multirocket
HAR(9)	0.67	0.53	0.74	**0.78**	0.92	**0.94**	0.94
EEG/MEG(6)	0.55	0.54	**0.58**	0.50	**0.55**	0.55	0.54
Audio Spectra(3)	**0.18**	0.16	0.18	0.18	0.46	**0.70**	0.52
Other(3)	0.52	0.58	0.65	**0.74**	**0.84**	0.83	0.80
ECG(2)	**0.46**	0.20	**0.46**	0.40	**0.27**	0.26	0.24
Motion(2)	0.59	0.65	0.65	**0.72**	0.99	**1.00**	**1.00**

6.2.3 Univariate Time Series Classification

In Figure 6.6, we compare the accuracy of four tabular models on univariate datasets: Random Forest, Logistic Regression, Ridge Regression (RidgeCV) and Linear Discriminant Analysis (LDA). The critical difference diagram [80] captures the average accuracy rank over all the datasets. The accuracy gain is evaluated using a Wilcoxon signed-rank test with Holm correction and visualised with the critical difference (CD) diagram with significance value (α) = 0.05. The figure illustrates Random Forest significantly outperforms the other three models and Logistic Regression outperforms the other linear models Table 6.5 illustrates the mean accuracy and total training and test computation time in minutes. The tabular results correspond to the tabular CD diagram, where Random Forest is the best classifier.

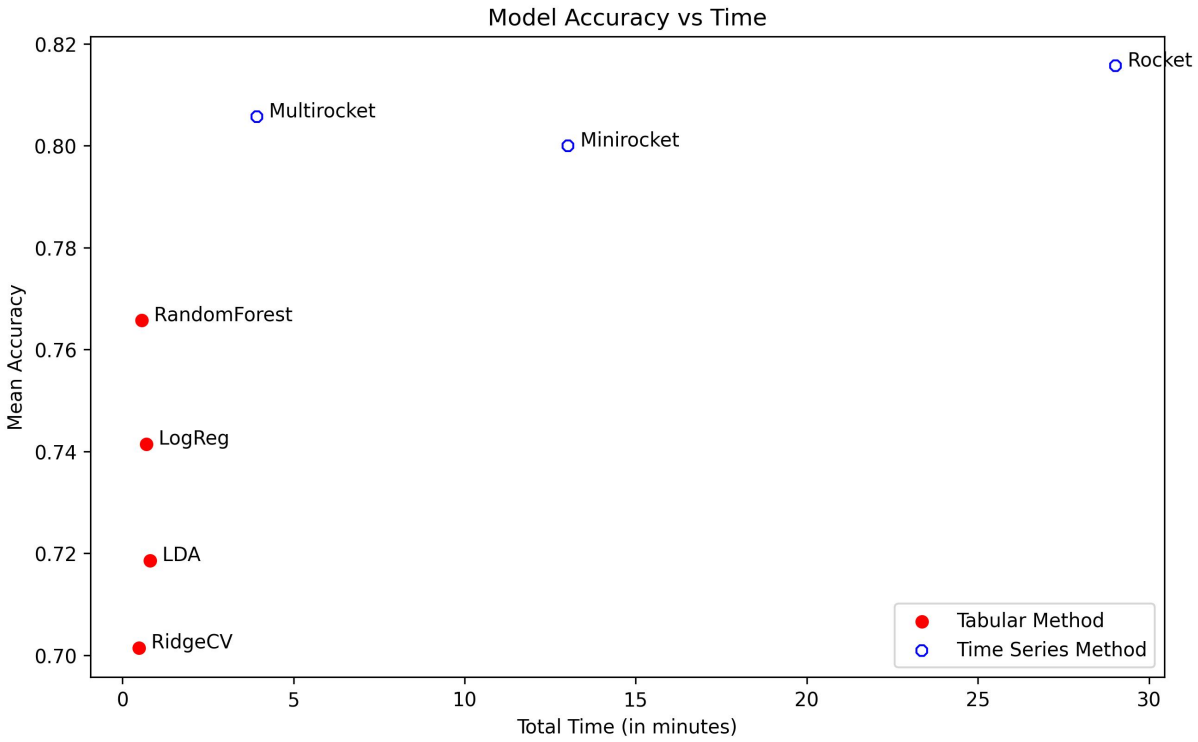


Figure 6.5: Accuracy-Time tradeoff for datasets in the **grey region** in Figure 6.4.

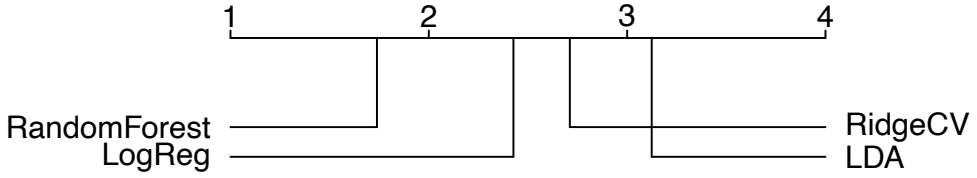


Figure 6.6: Accuracy comparison of tabular methods on UTSC datasets.

Table 6.5: Mean accuracy and total computation time taken by tabular models on UTSC datasets.

	Mean Accuracy	Total Time (minutes)
RandomForest	0.74	0.886
LogReg	0.69	0.31
RidgeCV	0.67	0.09
LDA	0.63	0.09

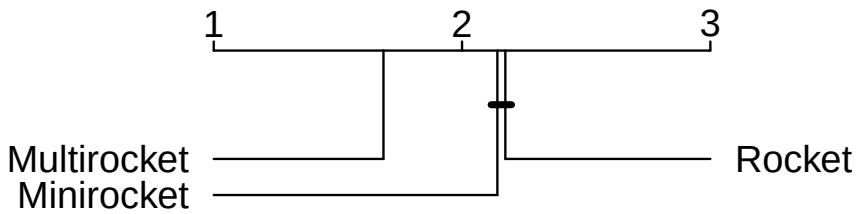


Figure 6.7: Accuracy comparison of time-series methods on UTSC datasets.

Table 6.6: Mean accuracy and total computation time taken by time-series models on UTSC datasets.

	Mean Accuracy	Total Time (minutes)
Minirocket	0.86	34.56
Multirocket	0.86	73.46
Rocket	0.85	158.76

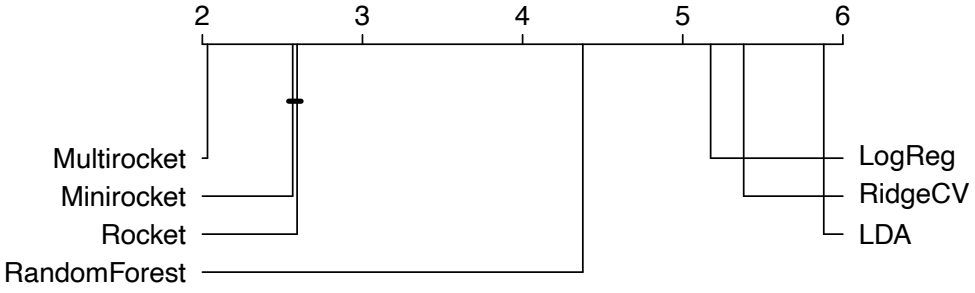


Figure 6.8: Accuracy comparison of tabular and time series models on UTSC datasets.

Time Series Methods Results. Similarly, in Figure 6.7 and Table 6.6, we compare the accuracy of three time series classification models: Multirocket, MiniRocket, and Rocket. We use the implementation in the aeon-toolkit library[2] with default parameters. From the critical difference diagram (Figure 6.7) we note that MultiRocket is significantly more accurate than MiniRocket and Rocket.

6.2.3.1 Time Series Methods vs Tabular Methods

In Figure 6.8, we compare the accuracy of time-series and tabular models. We can see that the time-series models have a higher mean accuracy rank than the tabular models. Multirocket is significantly more accurate than all other models, and Random Forest is the closest tabular model to the time-series models.

Figure 6.8 provides a summary overview of the performance of classifiers using their average accuracy ranking across the datasets analysed. Average behaviour with respect to accuracy or rank is a common and useful summary to get an overview of the performance of multiple classifiers over multiple datasets. However, it is crucial to examine the performance of models at a finer level to understand the difference in behaviour between tabular and time-series models.

For the UEA benchmark, surprisingly, 19.2% of the datasets performed better with tabular models (green region), 31.1% performed within 10 percentage points with both

[2] https://www.aeon-toolkit.org/en/latest/api_reference/classification.html

tabular and time series models (grey region), and 49.5% performed better than 10 percentage points with time series models (red region).

The above numbers imply that on about 19% of the benchmark, there are only weak temporal patterns, and tabular methods that disregard time ordering are very competitive when compared with time series methods. As a result, for many of those datasets in the green and grey region, using a complex time series model would be like using a sledgehammer to crack a nut. We of course acknowledge that time series methods work very well for the datasets in the red region, but these account for slightly less than half of the benchmark. We also acknowledge that the Rocket algorithms have been tested outside of this benchmark with good results in many real time series applications [81, 18, 82, 83]. The question remains though: should we include the datasets in the green and grey areas into a time series benchmark at all, given that tabular methods have similar accuracy to the best time series methods on those datasets.

6.2.3.2 Computation Time Analysis

Traditionally, tabular models are known for their computational speed. This is also evident from Tables 6.5 and 6.6, which show that tabular models are an order of magnitude faster than time series models. Figure 6.9 illustrates the various regions for accuracy, but it is worth highlighting that tabular models in the green and grey regions are faster and almost as accurate, or even more accurate than time series methods.

Figure 6.10 shows the tradeoff between the mean accuracy and total computation time for the various time-series and tabular models in grey region datasets. Multirocket and Random Forest are the most accurate models among time series and tabular models, respectively. The difference in accuracy between Multirocket and Random Forest is approximately 5 percentage points. However, Multirocket takes an average of 30 minutes longer to train.

6.2.3.3 Domain-wise Analysis

Table 6.7 shows the mean accuracy of different classifiers on datasets from various domains (as annotated by the meta-data in UCR/UEA). The benchmark is highly dominated by three domains: Image, Sensor, and Motion. About 63% of the benchmark comprises these three domains out of a total of 13 domains in the benchmark.

As expected, with regard to average accuracy in a specific domain, as also shown in Figure 6.8, time series models performed better than tabular models in most of the domains. However, we note that the tabular models performed especially well in the

Figure 6.9: Accuracy comparison of the best time series model with the best tabular model on univariate time series datasets. Red circles represent the tabular models, and blue circles represent the time series models. Each marker shows the maximum accuracy achieved by the tabular models versus the time series models.

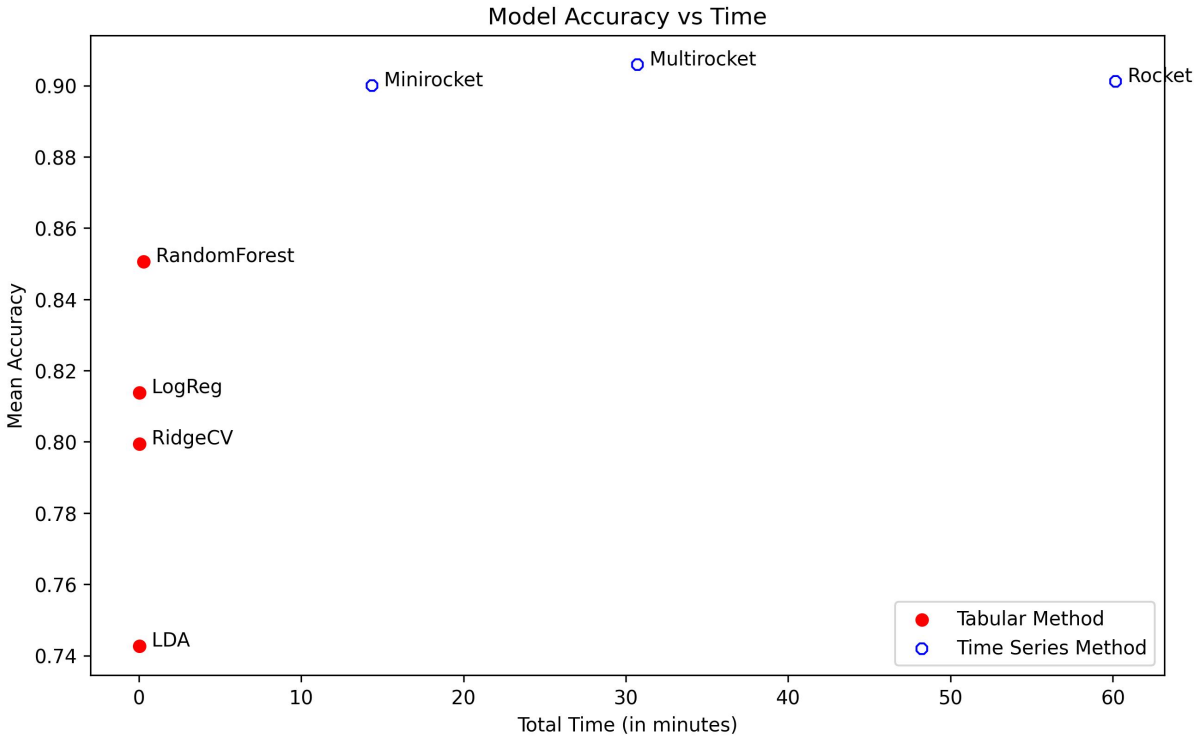


Figure 6.10: Accuracy-Time tradeoff for datasets in the **grey region** shown in Figure 6.9. We observe a mean accuracy difference of about 5 percentage points, but at least an order of magnitude difference in computation time, between tabular and time series methods.

Table 6.7: Mean accuracy of classifiers by problem types on UCR univariate datasets.

	Tabular Models				Time Series Models		
Domain (#datasets)	RidgeCV	LDA	LogReg	RandomForest	Rocket	Minirocket	Multirocket
Image(31)	0.66	0.62	**0.71**	0.75	0.85	0.85	**0.85**
Sensor(20)	0.73	0.69	0.72	**0.76**	0.86	0.86	**0.87**
Motion(17)	0.58	0.46	0.58	**0.70**	0.84	0.84	**0.85**
Device(8)	0.48	0.44	0.48	0.62	0.76	0.74	**0.77**
Simulated(8)	0.78	0.82	0.81	**0.88**	**0.99**	0.98	**0.99**
Spectro(8)	0.86	**0.90**	0.86	0.82	0.84	0.86	**0.86**
ECG(4)	**0.92**	0.84	0.92	0.82	0.97	0.97	**0.97**
Spectrum(4)	**0.75**	0.67	0.74	0.67	0.83	0.82	**0.88**
Hemodynamics(3)	0.05	**0.16**	0.12	0.13	0.66	**0.94**	0.81
EOG(2)	0.3	0.28	0.37	**0.43**	0.59	0.57	**0.60**
EPG(2)	0.82	**1.00**	**1.00**	1.00	0.99	1.00	1.00
Power(1)	0.98	0.73	0.99	**1.00**	0.92	**0.99**	0.98
Traffic(1)	**0.98**	0.95	0.98	0.98	0.98	0.98	**0.98**

Spectro domain. This could be because the Spectro domain does not have strong temporal features. Also, as we have seen in Figure 6.9, average behaviour can be misleading and we need to look at the accuracy on individual datasets to get a good idea of accuracy behaviour across the entire benchmark or specific domains.

6.2.4 Discussion and Lessons Learned

- **Redefining baselines:** Most previous research has considered 1NN-DTW as the baseline for time series classification. This is a reasonable choice, as 1NN-DTW is a simple and effective algorithm that is often competitive with more complex time series methods. However, our study suggests that simple tabular models can perform significantly well on some datasets, even when compared to recent state-of-the-art TSC algorithms. This finding suggests that there is a need to rethink how we do baseline comparisons for time series classification.

- **Not all that looks time series is a time series:** Our study demonstrated that tabular methods outperformed time series methods on some domains, specifically Spectro (Table 6.7), EEG or ECG (Table 6.4). This could be because the Spectro datasets did not contain strong temporal information. Either way, we need to ask whether it makes sense to have these datasets in a time series classification benchmark.

- **Considering trade-offs:** In our study we observed that time series models outperformed tabular models by a few percentage points on the red datasets. However, tabular models outperformed time series methods in the green datasets and were significantly faster to train and test. Therefore, especially for datasets in the grey region, where tabular and time series methods are close in accuracy, we recommend carefully considering whether tabular models are preferable to time series methods, especially if time is a constraint.

6.2.5 Improving Tabular Models

Since the above-mentioned experiments were conducted using the default hyperparameters, we wanted to investigate whether we could improve the performance of tabular models by tuning the hyperparameters. To do this, we performed hyperparameter tuning on Random Forest and Logistic Regression, since they were the best performing models in both univariate (Figure 6.10) and multivariate (Figure 6.5) experiments.

Table 6.8: Improvement on accuracy on univariate and multivariate datasets and mean computation time in minutes.

	Mean Accuracy		Mean Computation Time (minutes)	
	Before	After	Before	After
Univariate	0.86	0.87	0.47	13.41
Multivariate	0.74	0.75	0.91	43.10

We performed hyperparameter tuning with a combination of scaling and regularization. Table 6.8 shows the results of the hyperparameter tuning and the improvement for the best tabular model. We found that hyperparameter tuning can increase accuracy, but it also takes a significant amount of time to find the best hyperparameters.

6.3 Conclusion

In this study, we compared the performance of tabular models with state-of-the-art time series models on the UCR/UEA univariate and multivariate time series classification benchmarks. We found that tabular models performed surprisingly well on many datasets, outperforming the recent Multirocket classifier on a significant percentage of the datasets. On many other datasets, the accuracy was comparable, but tabular models were more efficient in terms of computation time. Overall, in about half of the datasets in either the univariate or the multivariate benchmarks, tabular methods were within 10 percentage points accuracy of the time series methods.

Our findings suggest that tabular models should be considered as baselines for evaluating improvements in time series classifiers, and even for considering whether a dataset should be included in the time series classification benchmarks. Furthermore, tabular methods can be a viable alternative to time series models for some classification tasks. Tabular models are easier to train and deploy, and they are more efficient in terms of computation time. The performance of tabular models does vary depending on the characteristics of the dataset. In future work, we plan to further investigate the factors that contribute to the performance of tabular models on time series data, and include more tabular models and parameter tuning.

CHAPTER

SEVEN

CONCLUSIONS

7.1 Contributions

Time Series Classification has gained considerable attention from researchers in recent years. However, the relentless pursuit of enhanced accuracy often diverts focus from crucial aspects such as scalability and sanity checks. Scalability is of paramount importance in time series classification for timely predictions and efficient infrastructure utilization. This holds true across various domains and fields. This thesis dived deep into exploration, analysis, and proposing solutions to enhance the scalability of MTSC. We delved into the classification process from multiple perspectives, identifying solutions to address MTSC's scalability challenges, particularly those related to time series length and handling a large number of channels.

With the rapid advancements in computing infrastructure, researchers often overlook fundamental checks, such as comparing newly proposed methods against traditional machine learning techniques like linear classifiers or other tabular approaches. These methods hold immense significance and should not be disregarded merely due to their established nature. On the contrary, they have been time-tested across various types of datasets and are often the primary methods deployed in real-world applications.

We summarize the key lessons learnt in this thesis:

- Prior to proposing a solution to any particular problem, it is crucial to examine the current state-of-the-art. In Chapter 3, we extensively examined various classifiers and evaluated the computation efficiency. We find that many classifiers are not statistically different from each other in accuracy but massively vary in training time. Additionally, some classifiers also have a bottleneck in memory usage; we attribute this to the fact that some of these classifiers were initially crafted to

deal with univariate time series and later adopted for multivariate time series classification.

- Additionally, in Chapter 3, we also showed methods to deal with time-series length. We illustrated that properties of time series can be captured with statistics, which can later be used to represent the entire time series (9_Mul_PAA). The result of the classification of MTS with feature vectors derived from the statistical values is not significantly different than the other SOTA classifiers, although the 9_Mul_PAA is much faster than classifiers like 1NN-DTW, WEASEL-MUSE and MrSEQL-SAX.

- Since most of the SOTA MTSC classifiers like ROCKET, WEASEL-MUSE, MrSEQL-SAX are adapted versions of their respective univariate time-series classifiers, they don't have the ability to deal with the number of channels in a multivariate time series. Either they concatenate all channels or treat each channel as a separate time series. The former case leads to a longer time series, while the latter may create confusion while assigning class labels. In Chapter 4, we proposed ECS and ECP, which are fast and effective methods for channel selection. We illustrated that with our channel selection methods, not only does the computation time of classification improve, but also, in some cases, we see improvement in classification accuracy.

- In Chapter 5, we build upon the channel selection from Chapter 4. We refined and improved the channel selection from the previous chapter; the class prototypes employed to perform channel selection were susceptible to noise. We proposed a better class prototype to enhance channel selection performance by dealing with outliers. Also, we evaluated the performance of channel selection methodologies in real-world and synthetic datasets.

- In Chapter 6, we took a step back in analysing the current state-of-the-art methods with tabular models. The results were quite intriguing; we found that on many datasets, the tabular models outperform classifiers like ROCKET, MiniROCKET and MultiROCKET, which are considered the best classifiers in the literature. We extended this study to univariate time series datasets as well, and we found similar and consistent results.

7.2 Future Work

The widespread availability of multivariate time series data has opened up a plethora of avenues for exploration, addressing pressing modern-day requirements and challenges. This section delves into potential challenges that could be addressed by building upon the work presented in this thesis. It is important to acknowledge that this list is not exhaustive, and there remain numerous other challenges that could be tackled.

7.2.1 Synthetic Datasets

With the growing interest in MTSC, more datasets are becoming available. However, there is still a scarcity of literature on generating synthetic datasets. MTSC datasets are characterized by the complexity of their channels, where different channels may or may not be dependent on each other partially or completely. The author of [3] attempted to address some of these challenges, but the author of [84] illustrated that even linear classifiers could perform exceptionally well on datasets generated from the study. This raises the question of whether these synthetically generated datasets exhibit any temporal component.

7.2.2 Extracting Shapelets

Similar to the channel selection techniques employed in our previous work, can we develop a scalable method for extracting important subsequences from time series data? While MrSEQL-SAX addresses this challenge to a certain extent, combining different channels introduces an additional layer of complexity. Shaplets from one channel may influence the other important subsequences in another channel, making it difficult to determine the relative importance of each subsequence. This issue necessitates a refined approach for extracting meaningful subsequences from multivariate time series data. The solution would not only improve the accuracy and scalability of classifiers but also could have a butterfly effect on research fields like causality and forecasting.

7.2.3 Noise Impact

Since the temporal data is mainly collected from different sensors, it is susceptible to noise due to errors being collected at the source. Therefore, there are methods required to factor in the noise; in our work, where we factored the noise into channel selection,

we produced some significant results on certain datasets. However, since the study of noise was not the primary objective of the thesis, we did not delve deep into the issue.

7.2.4 Better Channel Selection through Interaction

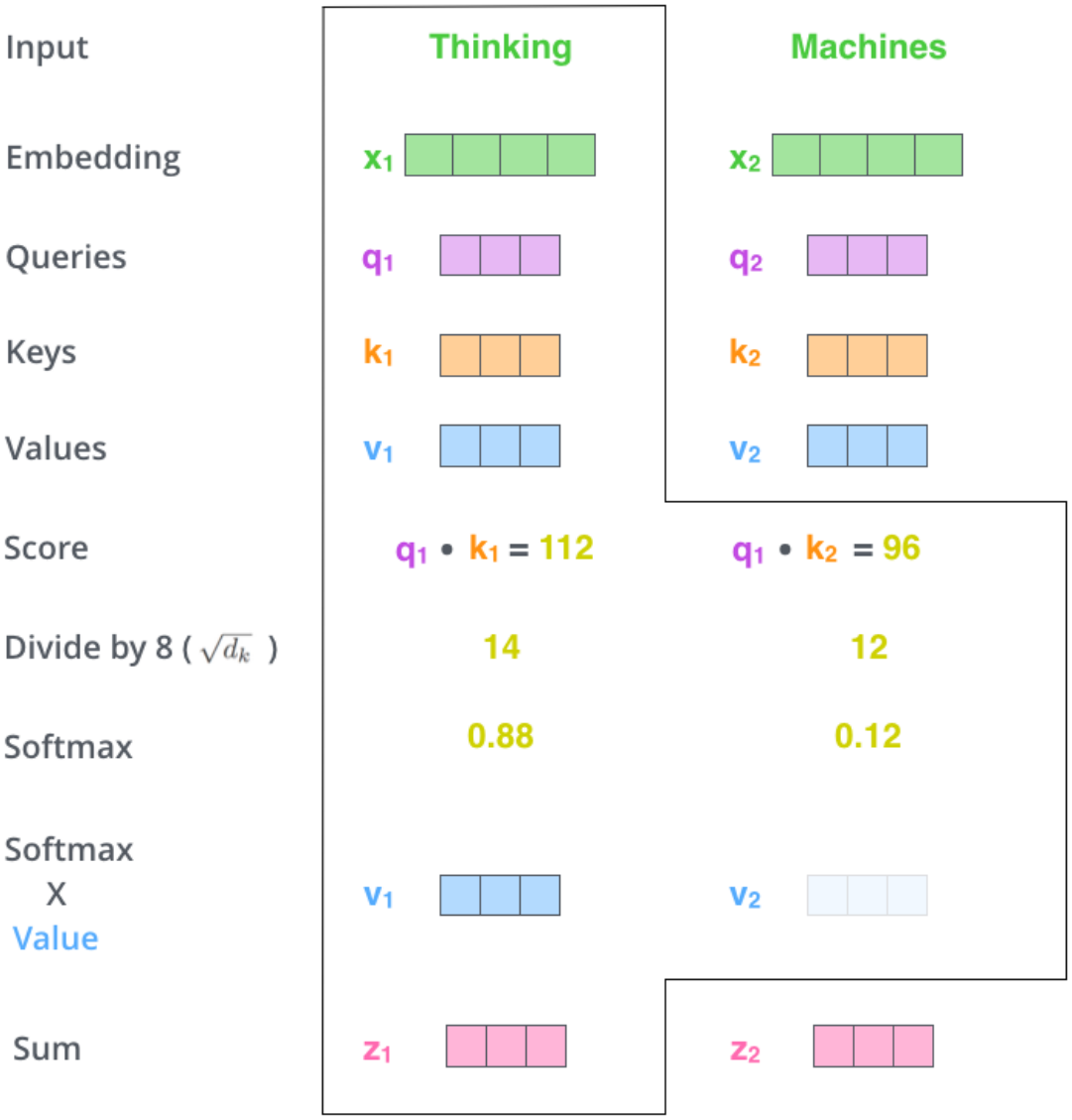


Figure 7.1: Image from The Illustrated Transformer blog [1].

Our study revealed that reducing redundant channels could significantly improve memory and computation efficiency. However, we have not yet investigated the relationships or interactions between channels. Furthermore, it is interesting to know how the behaviour changes when time series are hierarchical in nature. While we attempted to assess the impact of each channel on a specific class pair in Chapters 4 and 5,

[1] https://jalammar.github.io/illustrated-transformer

a more comprehensive analysis of the correlation structure among channels is needed. This could be achieved by reframing the multi-channel time series as a graph, where each node represents a channel and each edge depicts its relationship.

Another approach could involve interpreting the channels in a manner similar to words in the self-attention mechanism. For instance, as depicted in Figure 7.1, the output/sum z_1 of the attention mechanism incorporates information primarily from the word "Thinking" (approximately 88%) and a lesser degree from the word "Machines" (approximately 12%). This newly formed channel, or latent channel, could substitute the original channels. However, this method poses the challenge of interpretability in the latent space. Nevertheless, the success of the attention mechanism in various domains, including natural language processing (NLP), vision, and even forecasting, suggests that further exploration of its potential for channel selection/reduction is worthwhile.

7.2.5 Scalable Classifiers

In this thesis, we have proposed stand-alone methods, which are domain and classifier-agnostic. However, the existing literature on time series analysis needs specialized classifiers that can effectively handle the unique characteristics of different MTSC datasets, such as the length and number of channels. These characteristics play a critical role in the adaptability of classifiers, as different domains exhibit distinct behaviours. As mentioned, the current state-of-the-art (SOTA) classifiers, including the ROCKET family, MrSEQL-SAX, and WEASEL-MUSE, are adapted versions of their respective univariate time series (UTS) classifiers. The ROCKET family of classifiers employs random channel selection, while MrSEQL-SAX and WEASEL-MUSE consider data from all channels; this leaves a significant gap in the literature concerning the development of specialized MTS classifiers that can cater to the unique features of MTSC datasets. Recently, transformer-based models like ConvTran have been a step in that direction.

7.3 Final Discussion

This work started with the objective of developing methods for scalable time series classification. Initially, we aimed to create a specialized classifier for MTSC, but during the empirical evaluation in Chapter 3, we realized that focusing on individual bottlenecks, such as channel selection, would be a more effective approach. Although we achieved promising results, the work presented its own set of challenges.

One area where we invested significant time was noise removal from the datasets. We needed to ensure that the total computation time for channel selection and classification did not exceed that of the default classifier. To reduce noise, we experimented with various techniques, including Fast Fourier Transform, smoothing, and splines. We also developed a hypothesis to assess noise levels using an autoregressive approach. However, the initial attempts did not work well.

Another method to address noise was to use symbolic representation, which involves representing a subsequence with a word. However, converting to symbolic representation is computationally expensive, and we spent time optimizing this process.

Instead of using centroids, we explored the concept of shrunken centroids [85] for creating class prototypes. This approach showed promise in some datasets but introduced another hyperparameter to the problem.

Another direction we considered was identifying important subsequences using shapelets. We segmented the time series into small subsequences, trained a faster classifier like MiniROCKET on each segment, and then scored each segment individually and in combination with others. We selected the windows with the highest accuracy. However, we encountered computational limitations due to running multiple classifiers. Additionally, determining the optimal window length presented a challenge. Too small a window would lead to increased computation time, while too large a window could result in poor subsequences. Unfortunately, we had to limit this investigation due to time constraints.

While these challenges persist, this thesis successfully achieved its primary goal of scalable multivariate time series classification. We introduced novel methods for time series classification and provided the research community with valuable insights into the comparison of tabular and time series models. Our work on channel selection addressed a critical void and advanced scalability efforts, gaining positive recognition and integration into major open-source time series libraries. We embarked on a new frontier of scalability in multivariate time series classification, where previous advancements primarily focused on enhancing accuracy. Our findings demonstrate that accuracy and scalability can coexist and, in certain instances, complement each other.

Numerous exciting research directions can emerge from this work, as outlined in the preceding sections. We look forward to the practical application and evaluation of our findings in real-world settings.

7.4 Resources Made Publicly Available

We list here the resources that were used to generate the work of this thesis. We provide the Github links of all the chapters and other contributions made during the course of this thesis.

- Code for empirical evaluation of MTSC algorithms: https://github.com/mlgig/mtsc_benchmark
- Code for fast channel selection for scalable multivariate time series classification. https://github.com/mlgig/Channel-Selection-MTSC
- Code for scalable classifier-agnostic channel selection for multivariate time series classification: https://github.com/mlgig/ChannelSelectionMTSC
- Code for benchmarking the scalability accuracy trade-offs for tabular versus time series classification methods: https://github.com/mlgig/TabularModelsforTSC

Also, our code is available in open-source libraries like sktime[2] and aeon[3].

- Sktime: https://github.com/sktime/sktime/blob/main/examples/channel_selection.ipynb
- Aeon toolkit: https://www.aeon-toolkit.org/en/latest/examples/transformations/channel_selection.html

[2] https://github.com/sktime
[3] https://github.com/aeon-toolkit

BIBLIOGRAPHY

[1] Thach Le Nguyen, Severin Gsponer, Iulia Ilie, Martin O'Reilly, and Georgiana Ifrim. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data mining and knowledge discovery*, 33(4):1183–1222, 2019. vii, 7, 8

[2] Ashish Singh, Binh Thanh Le, Thach Le Nguyen, Darragh Whelan, Martin O'Reilly, Brian Caulfield, and Georgiana Ifrim. Interpretable classification of human exercise videos through pose estimation and multivariate time series analysis. In *5th International Workshop on Health Intelligence(W3PHIAI-21) at AAAI21*. Springer, 2021. vii, 1, 35

[3] Aya Abdelsalam Ismail, Mohamed Gunady, Hector Corrada Bravo, and Soheil Feizi. Benchmarking deep learning interpretability in time series predictions. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6441–6452. Curran Associates, Inc., 2020. viii, 71, 72, 97

[4] Zahra Karevan and Johan AK Suykens. Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9, 2020. 1

[5] Young Shin Kim, Svetlozar T Rachev, Michele Leonardo Bianchi, Ivan Mitov, and Frank J Fabozzi. Time series analysis for financial market meltdowns. *Journal of Banking & Finance*, 35(8):1879–1891, 2011. 1

[6] Shen Yin, Steven X Ding, Xiaochen Xie, and Hao Luo. A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial electronics*, 61(11):6418–6428, 2014. 1

[7] Charlie Osborne. Covid-19 a 'significant' factor in wearable device adoption, market surge, 2023. 1

[8] Mordor Intelligence. Wearable technology market size & share analysis - growth trends & forecasts (2023 - 2028), 2023. 1

[9] Colin Adams, Luis Alonso, Benjamin Atkin, John Banning, Sumeer Bhola, Rick Buskens, Ming Chen, Xi Chen, Yoo Chung, Qin Jia, Nick Sakharov, George Talbot, Nick Taylor, and Adam Tart. Monarch: Google's planet-scale in-memory time series database. *Proc. VLDB Endow.*, 13(12):3181–3194, 2020. 1

[10] L Riaboff, L Shalloo, AF Smeaton, S Couvreur, A Madouasse, and MT Keane. Predicting livestock behaviour using accelerometers: A systematic review of processing techniques for ruminant behaviour prediction from raw accelerometer data. *Computers and Electronics in Agriculture*, 192:106610, 2022. 1

[11] Marcos Fabietti, Mufti Mahmud, and Ahmad Lotfi. On-chip machine learning for portable systems: Application to electroencephalography-based brain-computer interfaces. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. 1

[12] 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 Mining and Knowledge Discovery*, 31:606–660, 2017. 2, 39, 60

[13] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018. 2, 8, 18, 19, 25, 27

[14] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978. 2, 78

[15] Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+ muse. *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD18), arXiv preprint arXiv:1711.11343*, 2017. 2

[16] Bhaskar Dhariyal, Thach Le Nguyen, Severin Gsponer, and Georgiana Ifrim. An examination of the state-of-the-art for multivariate time series classification. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 243–250, 2020. 3

[17] Bhaskar Dhariyal, Thach Le Nguyen, and Georgiana Ifrim. Fast channel selection for scalable multivariate time series classification. In *International Workshop on Advanced Analytics and Learning on Temporal Data*, pages 36–54. Springer, 2021. 4, 16, 51, 53

[18] Bhaskar Dhariyal, Thach Le Nguyen, and Georgiana Ifrim. Scalable classifier-agnostic channel selection for multivariate time series classification. *Data Min. Knowl. Discov.*, 37(2):1010–1054, 2023. 4, 90

[19] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 248–257. ACM, 2021. 5, 12

[20] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I Webb. Multirocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36(5):1623–1646, 2022. 5, 12, 78, 79

[21] Bhaskar Dhariyal, Thach Le Nguyen, and Georgiana Ifrim. Back to basics: A sanity check on modern time series classification algorithms. In Georgiana Ifrim, Romain Tavenard, Anthony Bagnall, Patrick Schaefer, Simon Malinowski, Thomas Guyet, and Vincent Lemaire, editors, *Advanced Analytics and Learning on Temporal Data*, pages 205–229, Cham, 2023. Springer Nature Switzerland. 5

[22] Mohammad Shokoohi-Yekta, Jiaoling Wang, and Eamonn J. Keogh. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *SDM*, 2015. 9

[23] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, pages 1–49, 2020. 9, 14, 34

[24] Thach Le Nguyen, Severin Gsponer, Iulia Ilie, Martin O'Reilly, and Georgiana Ifrim. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data Mining and Knowledge Discovery*, 33(4):1183–1222, Jul 2019. 9, 19, 34

[25] Georgiana Ifrim and Carsten Wiuf. Bounded coordinate-descent for biological sequence classification in high dimensional predictor space. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 708–716, New York, NY, USA, 2011. Association for Computing Machinery. 9

[26] Thach Le Nguyen, Severin Gsponer, and Georgiana Ifrim. Time series classification by sequence learning in all-subsequence space. In *2017 IEEE 33rd international conference on data engineering (ICDE)*, pages 947–958. IEEE, 2017. 9

[27] Patrick Schäfer and Mikael 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*, pages 516–527, 2012. 9

[28] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, Oct 2007. 9

[29] Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+muse. *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD18), arXiv preprint arXiv:1711.11343*, 2018. 11, 34

[30] Patrick Schäfer and Ulf Leser. Fast and accurate time series classification with WEASEL. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 637–646. ACM, 2017. 11

[31] Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.*, 34(5):1454–1495, 2020. 12, 19, 20, 34, 78, 79

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 12

[33] Xizhe Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In *AAAI*, 2020. 13, 19

[34] Houshang Darabi Fazle Karim, Somshubra Majumdar and Samuel Harford. Multivariate lstm-fcns for time series classification. abs/1801.04503, 2018. 13

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 13

[36] Navid Mohammadi Foumani, Chang Wei Tan, Geoffrey I Webb, and Mahsa Salehi. Improving position encoding of transformers for multivariate time series classification. *arXiv preprint arXiv:2305.16642*, 2023. 14

[37] Shuchu Han and Alexandru Niculescu-Mizil. Supervised feature subset selection and feature ranking for multivariate time series without feature extraction. *arXiv preprint arXiv:2005.00259*, 2020. 14, 16

[38] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2114–2124, 2021. 14

[39] Demetres Kostas, Stephane Aroca-Ouellette, and Frank Rudzicz. Bendr: using transformers and a contrastive self-supervised learning task to learn from massive amounts of eeg data. *Frontiers in Human Neuroscience*, 15:653659, 2021. 14

[40] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, Mar 2005. 14

[41] J. Lines, S. Taylor, and A. Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1041–1046, Dec 2016. 14

[42] Matthew Middlehurst, James Large, and Anthony Bagnall. The canonical interval forest (cif) classifier for time series classification. In *2020 IEEE international conference on big data (big data)*, pages 188–195. IEEE, 2020. 14

[43] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. Generalized random shapelet forests. *Data mining and knowledge discovery*, 30:1053–1085, 2016. 14

[44] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Min. Knowl. Discov.*, 34(6):1936–1962, 2020. 14, 78

[45] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11):3211–3243, Dec 2021. 14, 16

[46] George H John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine learning proceedings 1994*, pages 121–129. Elsevier, 1994. 15, 61

[47] Bahavathy Kathirgamanathan and Pádraig Cunningham. A feature selection method for multi-dimension time-series data. In *International Workshop on Advanced Analytics and Learning on Temporal Data*, pages 220–231. Springer, 2020. 15, 16, 61

[48] Bahavathy Kathirgamanathan, Cillian Buckley, Brian Caulfield, and Pádraig Cunningham. Feature subset selection for detecting fatigue in runners using time series sensor data. In Mounîm El Yacoubi, Eric Granger, Pong Chi Yuen, Umapada Pal, and Nicole Vincent, editors, *Pattern Recognition and Artificial Intelligence*, pages 541–552, Cham, 2022. Springer International Publishing. 15

[49] Hyunjin Yoon, Kiyoung Yang, and Cyrus Shahabi. Feature subset selection and feature ranking for multivariate time series. *IEEE transactions on knowledge and data engineering*, 17(9):1186–1198, 2005. 15

[50] WJ Krzanowski. Between-groups comparison of principal components. *Journal of the American Statistical Association*, 74(367):703–707, 1979. 15

[51] Bing Hu, Yanping Chen, Jesin Zakaria, Liudmila Ulanova, and Eamonn Keogh. Classification of multi-dimensional streaming time series by weighting each classifier's track record. In *2013 IEEE 13th International Conference on Data Mining*, pages 281–290, 2013. 16

[52] Alejandro Pasos Ruiz and Anthony Bagnall. Dimension selection strategies for multivariate time series classification with hive-cotev2. 0. In *International Workshop on Advanced Analytics and Learning on Temporal Data*, pages 133–147. Springer, 2022. 16, 49

[53] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019. 19

[54] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 20

[55] Carl Henning Lubba, Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. catch22: Canonical time-series characteristics selected through highly comparative time-series analysis. *Data Mining and Knowledge Discovery*, 33:1821â1852, 2019. 21

[56] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001. 21

[57] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, December 2006. 25, 60

[58] Salvador Garcia and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 12 2008. 25, 60

[59] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *Journal of Machine Learning Research*, 17(5):1–10, 2016. 25, 60

[60] Borja Calvo and Guzman Santafe. scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. *The R Journal*, 8(1):248–256, 2016. 25, 60

[61] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, 2021. 28

[62] Mark Rippetoe. The press. *Cross Fit Journal*, (45), 2006. 34

[63] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011. 37

[64] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019. 39, 59

[65] K Sainio, M-L Grandström, O Pettay, and M Donner. Eeg in neonatal herpes simplex encephalitis. *Electroencephalography and clinical neurophysiology*, 56(6):556–561, 1983. 50

[66] Ray Yeutien Chou. Forecasting financial volatilities with extreme values: the conditional autoregressive range (carr) model. *Journal of Money, Credit and Banking*, pages 561–582, 2005. 50

[67] Luis David Avendaño-Valencia, Eleni N Chatzi, Ki Young Koo, and James MW Brownjohn. Gaussian process time-series models for structures under operational variability. *Frontiers in Built Environment*, 3:69, 2017. 50

[68] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, 49(4):764–766, 2013. 52

[69] Christian Perwass, Herbert Edelsbrunner, Leif Kobbelt, and Konrad Polthier. *Geometric algebra with applications in engineering*, volume 4. Springer, 2009. 57

[70] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003. 61

[71] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 248–257, 2021. 78, 79

[72] Alan Julian Izenman. Linear discriminant analysis. In *Modern multivariate statistical techniques: regression, classification, and manifold learning*, pages 237–280. Springer, 2013. 78

[73] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958. 78

[74] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, pages 80–86, 2000. 78

[75] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001. 78

[76] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023. 78

[77] Maria Frizzarin, Giulio Visentin, Alessandro Ferragina, Elena Hayes, Antonio Bevilacqua, Bhaskar Dhariyal, Katarina Domijan, Hussain Khan, Georgiana Ifrim, Thach Le Nguyen, et al. Classification of cow diet based on milk mid infrared spectra: A data analysis competition at the âinternational workshop on spectroscopy and chemometrics 2022â. *Chemometrics and Intelligent Laboratory Systems*, page 104755, 2023. 78, 84

[78] Maria Frizzarin, Antonio Bevilacqua, Bhaskar Dhariyal, Katarina Domijan, Federico Ferraccioli, Elena Hayes, Georgiana Ifrim, Agnieszka Konkolewska, Thach Le Nguyen, Uche Mbaka, Giovanna Ranzato, Ashish Singh, Marco Stefanucci, and Alessandro Casa. Mid infrared spectroscopy and milk quality traits: A data analysis competition at the âinternational workshop on spectroscopy and chemometrics 2021â. *Chemometrics and Intelligent Laboratory Systems*, 219:104442, 2021. 78

[79] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019. 79

[80] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006. 87

[81] Ashish Singh, Antonio Bevilacqua, Thach Le Nguyen, Feiyan Hu, Kevin McGuinness, Martin O'Reilly, Darragh Whelan, Brian Caulfield, and Georgiana Ifrim. Fast and robust video-based exercise classification via body pose tracking and scalable multivariate time series classifiers. *Data Min. Knowl. Discov.*, 37(2):873–912, 2023. 90

[82] Anthony J. Bagnall, Aaron Bostrom, James Large, and Jason Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version. *CoRR*, abs/1602.01711, 2016. 90

[83] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony J. Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.*, 35(2):401–449, 2021. 90

[84] Davide Italo Serramazza, Thu Trang Nguyen, Thach Le Nguyen, and Georgiana Ifrim. Evaluating explanation methods for multivariate time series classification. pages 159–175, 2023. 97

[85] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, 2002. 100