Vincent Lemaire · Georgiana Ifrim ·
Anthony Bagnall · Thomas Guyet ·
Simon Malinowski · Patrick Schäfer ·
Romain Tavenard (Eds.)

Advanced Analytics and Learning on Temporal Data

LNAI 15433

9th ECML PKDD Workshop, AALTD 2024
Vilnius, Lithuania, September 9–13, 2024
Revised Selected Papers

Springer

Lecture Notes in Computer Science

Lecture Notes in Artificial Intelligence 15433

Founding Editor

Jörg Siekmann

Series Editors

The series Lecture Notes in Artificial Intelligence (LNAI) was established in 1988 as a topical subseries of LNCS devoted to artificial intelligence.

The series publishes state-of-the-art research results at a high level. As with the LNCS mother series, the mission of the series is to serve the international R & D community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings.

Vincent Lemaire · Georgiana Ifrim ·
Anthony Bagnall · Thomas Guyet ·
Simon Malinowski · Patrick Schäfer ·
Romain Tavenard
Editors

Advanced Analytics and Learning on Temporal Data

9th ECML PKDD Workshop, AALTD 2024
Vilnius, Lithuania, September 9–13, 2024
Revised Selected Papers

Springer

Editors
Vincent Lemaire
Orange Labs
Lannion, France

Anthony Bagnall
School of Electronics and Computer Science
University of Southampton
Southampton, UK

Simon Malinowski
University of Rennes
Rennes Cedex, France

Romain Tavenard
Université de Rennes 2
Rennes, France

Georgiana Ifrim
School of Computer Science
University College Dublin
Dublin, Ireland

Thomas Guyet
Université Claude Bernard Lyon 1
Villeurbanne, France

Patrick Schäfer
Department of Computer Science
Humboldt University of Berlin
Berlin, Germany

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-031-77065-4 ISBN 978-3-031-77066-1 (eBook)
https://doi.org/10.1007/978-3-031-77066-1

LNCS Sublibrary: SL7 – Artificial Intelligence

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Workshop Description

The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) is the premier European machine learning and data mining conference and builds upon over 22 years of successful events and conferences held across Europe. This year, ECML PKDD 2024 took place in Vilnius, Lithuania, during September 9–13, 2024. The main conference was complemented by a workshop program, where each workshop was dedicated to specialized topics, cross-cutting issues, and upcoming research trends. This standalone LNAI volume includes the selected papers of the 9th International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD), held at ECML PKDD 2024.

Motivation – Temporal data are frequently encountered in a wide range of domains such as bioinformatics, medicine, finance, environment, and engineering, among many others. They are naturally present in emerging applications such as motion analysis, energy efficient buildings, smart cities, social media, or sensor networks. Contrary to static data, temporal data are of a complex nature: they are generally noisy and of high dimensionality, they may be non-stationary (i.e., first-order statistics vary with time) and irregular (i.e., involving several time granularities), and they may have several invariant domain-dependent factors such as time delay, translation, scale, or tendency effects. These temporal peculiarities limit the majority of standard statistical models and machine learning approaches, Which mainly assume i.i.d data, homoscedasticity, normality of residuals, etc. To tackle such challenging temporal data we require new advanced approaches at the intersection of statistics, time series analysis, signal processing, and machine learning. Defining new approaches that transcend boundaries between several domains to extract valuable information from temporal data is undeniably an important research topic that has been the subject of active research in the last decade and will continue to be so for the foreseeable future.

Workshop Topics – The aim of the AALTD[1] workshop series is to bring together researchers and experts in machine learning, data mining, pattern analysis, and statistics to share their challenges and advances in temporal data analysis. Analysis and learning from temporal data covers a wide scope of tasks including metric learning, representation learning, unsupervised feature extraction, clustering, and classification.

For this ninth edition, the workshop received papers that cover one or several of the following topics:

- Temporal data clustering
- Classification and regression of univariate and multivariate time series
- Early classification of temporal data
- Deep learning for temporal data

[1] https://ecml-aaltd.github.io/aaltd2024/.

- Learning representation for temporal data
- Metric and kernel learning for temporal data
- Modeling temporal dependencies
- Time series forecasting
- Time series annotation, segmentation, and anomaly detection
- Spacial-temporal statistical analysis
- Functional data analysis methods
- Data streams
- Interpretable/explainable time-series analysis methods
- Dimensionality reduction, sparsity, algorithmic complexity, and big data challenges
- Benchmarking and assessment methods for temporal data
- Applications, including bioinformatics, medical, and energy consumption on temporal data

We also welcomed contributions that addressed aspects including, but not limited to, novel techniques, innovative applications, and techniques for the use of hybrid models.

Outcomes – AALTD 2024 was structured as a full-day workshop. We encouraged submissions of regular papers that were up to 16 pages of previously unpublished work. All submitted papers were peer-reviewed (double-blind) by two or three reviewers from the Program Committee, and selected on the basis of these reviews. AALTD 2024 received 15 submissions, among which 8 papers were accepted for inclusion in the proceedings. All the papers were invited for oral presentation and authors were given the opportunity to also present a poster through a discussion session. The workshop had an invited talk on "Time Series Anomaly Detection (TSAD)"[2] given by Paul Boniol[3], a researcher at Inria and a member of the VALDA project team which is a joint team between Inria Paris, École Normale Supérieure, and CNRS, France

We thank all the authors, reviewers, and organizers for the time and effort invested to make this workshop a success. We would also like to express our gratitude to the members of the Program Committee, the Organizing Committee of ECML PKDD 2024 and the local technical staff who helped us make AALTD 2024 a successful workshop. Sincere thanks are due to Springer for their help in publishing the proceedings. Lastly, we thank all the participants and speakers at AALTD 2024 for their contributions. Their collective support made the workshop an exciting, interesting and successful event.

October 2024

Vincent Lemaire
Georgiana Ifrim
Anthony Bagnall
Thomas Guyet
Simon Malinowski
Patrick Schäfer
Romain Tavenard

[2] https://ecml-aaltd.github.io/aaltd2024/invitedtalk.html.
[3] https://boniolp.github.io/.

Organization

Program Committee Chairs

Anthony Bagnall	University of Southampton, UK
Thomas Guyet	Inria, France
Georgiana Ifrim	University College Dublin, Ireland
Vincent Lemaire	Orange Innovation, France
Simon Malinowski	Université de Rennes, Inria, CNRS, IRISA, France
Patrick Schäfer	Humboldt-Universität zu Berlin, Germany
Romain Tavenard	Université de Rennes 2, France

Program Committee

Gustau Camps-Valls	Universitat de València, Spain
Pádraig Cunningham	University College Dublin, Ireland
Antoine Cornuéjols	AgroParisTech, France
Rémi Emonet	Jean Monnet University of Saint-Étienne, France
Germain Forestier	University of Upper Alsace, France
David Guijo-Rubio	Universidad de Córdoba, Spain
Davide Italo Serramazza	University College Dublin, Ireland
Florian Kalinke	Karlsruhe Institute of Technology, Germany
Colin Leverger	Orange Innovation, France
Thach Le Nguyen	University College Dublin, Ireland
Allan Tucker	Brunel University, UK

Contents

Conformal Prediction Techniques for Electricity Price Forecasting

Ciaran O'Connor[1(✉)], Steven Prestwich[2], and Andrea Visentin[2]

[1] SFI CRT in Artificial Intelligence, School of Computer Science & IT, University College Cork, Cork, Ireland
119226305@umail.ucc.ie

[2] Insight Centre for Data Analytics, School of Computer Science & IT, University College Cork, Cork, Ireland
{s.prestwich,andrea.visentin}@insight-centre.org

Abstract. Integrating the erratic production of renewable energy into the electricity grid poses numerous challenges. One approach to stabilising market prices and reducing energy losses due to curtailments is the deployment of batteries. Efficient electricity arbitrage is crucial to make investments in storage systems financially viable; trading solutions to achieve this rely on price forecasting techniques. This study delves into the application of Conformal Prediction (CP) techniques, including Ensemble Batch Prediction Intervals (EnbPI) and Sequential Predictive Conformal Inference for Time Series (SPCI), for generating probabilistic forecasts in the Irish electricity market. Recent advancements in CP have addressed temporal considerations inherent in time series forecasting, eliminating the need for exchangeability assumptions. Our study demonstrates that despite potential efficiency trade-offs, CP methods consistently yield precise and reliable prediction intervals, ensuring comprehensive coverage. We assess the impact of CP on the financial results of a simulated trading algorithm. Monetary outcomes achieved with EnbPI and SPCI outperform those of both split CP and traditional quantile regression models, highlighting the practical superiority of CP in electricity price forecasting.

1 Introduction

Electricity price forecasting (EPF) is paramount for energy companies navigating volatile markets, sudden price shifts, and changing demand patterns. The widespread integration of renewables can introduce volatility in net power supply due to rapid and unforeseen changes in their output, potentially resulting in reliability concerns within the power system (Martinez-Anido et al. (2016)). The incorporation of energy storage technologies such as Battery Energy Storage Systems (BESS) can enhance the reliability and efficiency of the grid, improving market liquidity and reducing price volatility. With the integration of renewable energy sources and smart grids, forecasting accuracy becomes increasingly critical. Accurate predictions stabilize energy production planning and inform

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 1–17, 2025.
https://doi.org/10.1007/978-3-031-77066-1_1

risk-aware strategies. Ireland is particularly interesting in this perspective. The renewable component accounted for 39.5% of the total electricity production in 2020 (EirGrid (2022)). However, a consistent part of this power is wasted due to curtailments, e.g. when the production exceeds the demand. In Ireland, between 3% to 6% of electricity is lost every year due to this offer/demand mismatch. This barrier strongly limits the extension of the current renewable production and hinders the efforts to reach carbon neutrality. The introduction of BESS is an efficient way to tackle this issue. Precise forecasts are indispensable for market operations, notably in the Day-Ahead Market (DAM), Intra-Day Market (IDM), and Balancing Market (BM) within European electricity markets (Green and Vasilakos (2010), Martinez-Anido et al. (2016)). In the Irish single electricity market, the DAM represents a market with significant contributions to grid stability, with volume far exceeding both the IDM and BM. The DAM facilitates trading for electricity delivery the next day, with daily auctions at noon CET that establish initial market prices for electricity. The transition from deterministic to probabilistic forecasting signifies a significant shift in the need for nuanced predictions in the face of escalating uncertainties in future supply, demand, and prices. Probabilistic Electricity Price Forecasting (PEPF), in particular quantile forecasts, has emerged to address uncertainties and provide decision-makers with a comprehensive understanding of potential outcomes. While traditional point forecasting methods established the groundwork, the rise of quantile forecasting, spurred by initiatives like the Global Energy Forecasting Competition 2014 (GEFCom2014), has ushered in a new era. However, the challenge of robust uncertainty quantification persists, prompting exploration beyond conventional methodologies such as Quantile Regression (QR) and QR Averaging (QRA) (Maciejowska et al. (2016), Nowotarski and Weron (2018), Uniejewski and Weron (2021)).

Conformal Prediction (CP) emerges as a promising alternative to both QR and QRA for generating prediction intervals (PI), offering both validity and adaptability without relying on strict assumptions. Initially introduced in Gammerman et al. (1998) and subsequently extended to regression and classification domains by Vovk et al. (2005) and Shafer and Vovk (2008). CP works by using past data to create a model that predicts future outcomes, providing a measure of confidence in the PI. A key feature of CP is its ability to deliver valid PI regardless of the underlying model, making it a flexible and reliable tool for uncertainty quantification. One of the foundational concepts in CP is data exchangeability, which assumes that the order of data points does not affect the statistical properties of the data set. While this assumption simplifies the development of CP methods, it presents significant challenges in time series applications, where the order and dependencies of data points are crucial. Traditional CP methods, which depend on exchangeability, often struggle to handle these dependencies effectively. To address these limitations, recent advancements in CP, such as Ensemble Batch Prediction Intervals (EnbPI) (Xu and Xie (2021) and Sequential Predictive Conformal Inference for Time Series (SPCI) (Xu and Xie (2023), have been developed. EnbPI enhances the flexibility and accuracy of PI by using

ensemble methods, which combine multiple models to improve predictive performance. SPCI, on the other hand, modifies the CP framework to better handle the sequential nature of time series data, ensuring that the PI remain valid even when data points are dependent on previous values. Focusing on these novel methods is important because they significantly improve the applicability of CP in dynamic fields like electricity markets, where accurate and reliable predictions are crucial for decision-making. By overcoming CP's traditional limitations, EnbPI and SPCI provide more reliable PI, crucial for renewable energy integration, where high uncertainties make reliable forecasts essential for market stability and efficient trading. This paper undertakes a thorough investigation into PEPF by leveraging recent advancements in CP methodologies, particularly adaptations tailored for time series data. We examine these adapted CP techniques alongside traditional QR approaches. Our study focuses on recent contributions to probabilistic forecasting, emphasizing how uncertainty can be transformed into an opportunity rather than a source of risk. In an extensive numerical analysis, the significance of coverage guarantees in enhancing trading strategies and fostering market resilience is assessed with an economic simulation.

The structure of this paper is as follows: Sect. 2 provides an overview of recent advancements in PEPF within the context of the DAM. Section 3 presents the dataset used in our empirical analysis. In Sect. 4, we detail our methodological framework, including our approach, models, and trading strategies. Section 5 presents the empirical findings, comparing the efficacy of CP with traditional methods through quantitative metrics and financial evaluations. Finally, Sect. 6 summarizes our findings.

2 Related Work

In this section, we present an overview of recent advancements in PEPF methodologies, focusing on modern CP techniques within the context of their application to the DAM.

2.1 Probabilistic Forecasting

Recent reviews by Khosravi and Nahavandi (2014) and Khajeh and Laaksonen (2022) have underscored the growing prominence of probabilistic forecasting in addressing uncertainties inherent in smart grids, supply-demand dynamics, and price variations. Notably, Nowotarski and Weron (2018) and Tzallas et al. (2022) have provided extensive insights into various PEPF methodologies, ranging from autoregressive models to neural networks and ensemble techniques. These reviews have critically evaluated methods like QRA, highlighting its dominance despite inherent limitations. Recent advancements in prediction interval generation, such as the methodology proposed in S. Salem et al. (2020), which leverages ensemble neural networks to generate prediction intervals alongside point estimates, contribute to the evolving landscape of regression analysis. Leverger et al. (2021) introduces a method that uses clustering to identify seasonal

patterns and classification to enhance forecast accuracy. This hybrid approach improves the reliability of probabilistic forecasts for seasonal data. Furthermore, Oesterheld et al. (2023) delves into the realm of performative predictions, where the act of making predictions can influence outcomes, shedding light on a crucial aspect of predictive modelling. In response to these limitations, recent studies by Uniejewski and Weron (2021), Uniejewski (2023) have introduced novel approaches such as Lasso QRA and smoothed QRA with kernel estimation, showcasing superior performance in trading strategies and financial outcomes. O'Connor et al. (2024) compares statistical, machine learning, and deep learning models for EPF in the Irish BM, finding that simpler statistical models like LEAR outperform more complex ones. The study provides a framework for model evaluation and offers an open-source dataset and models, which we utilize for our forecasting evaluation. Additionally, advancements in deep learning models, as explored in Lago et al. (2021), (2018), Marcjasz et al. (2020), (2022), have demonstrated notable improvements in both point and probabilistic forecasting.

2.2 Conformal Prediction

CP has emerged as a versatile framework for uncertainty quantification, as highlighted by (2023), emphasizing its significance in regression prediction intervals. Notably, recent work in (2022) has addressed CP's traditional challenge in handling time series data by introducing weighted residual distributions, enhancing robustness and reliability in prediction intervals, while (2023) presents an approach to improve CP's robustness to outliers. In the domain of time series forecasting, Dewolf et al. (2022) introduced Ensemble Conformalized Quantile Regression (EnCQR), showcasing superior performance in handling heteroscedastic data and ensuring reliable prediction intervals. Similarly, Foygel Barber et al. (2022) employed Conformal Quantile Regression (CQR) with neural networks, surpassing traditional methods in wind power forecasting accuracy. In a PEPF context, the only paper of its kind, Kath and Ziel (2021) looks at CP as a robust framework to enhance time series forecasting and manage uncertainty. CP provides dynamic and symmetric prediction intervals, emphasizing balanced construction, effective sampling, and error-based normalization. Comparative analysis with QRA underscores CP's market sensitivity and model selection importance. Of particular relevance to our study, Kath and Ziel (2021) investigated CP as a framework to enhance time series forecasting and manage uncertainty. Recent advancements like EnbPI proposed by Xu and Xie (2021) and SPCI introduced by Xu and Xie (2023) have addressed exchangeability issues in time series data, offering promising avenues for uncertainty estimation in PEPF applications.

2.3 Trading

In the realm of energy trading, Krishnamurthy et al. (2017), l Narajewski and Ziel (2021), Uniejewski and Weron (2021), Uniejewski (2023) have made significant contributions to the development of effective trading strategies for energy

storage systems. Noteworthy findings include the supremacy of stochastic models and optimal bidding strategies. Additionally, studies by Staffell and Rustomji (2016), Tohidi and Gibescu (2019), and Abramova and Bunn (2021) have explored the economic viability of energy storage systems, addressing various aspects such as profitability, revenue evaluation, and battery pack degradation. O'Connor et al. (2024b) integrates renewable energy into markets with battery storage. The study enhances DAM and BM trading using quantile-based forecasts and increased trading frequency. It highlights the economic viability of larger batteries, precise quantile pair selection, and high-frequency trading for maximizing profits. In summary, this review highlights advancements in PEPF methodologies, encompassing DAM forecasting, CP integration, insights into energy storage systems, and effective trading strategies. However, notable gaps remain, particularly regarding the applicability of recent improvements in probabilistic approaches to EPF. Our study aims to address these gaps by presenting methodologies to enhance energy trading strategies in the DAM, focusing on modern adaptations of CP for time series applications.

3 Datasets

Data for this study were sourced from the Single Electricity Market Operator for Ireland (SEMO). Information regarding prices, network parameters and forecasts are available from SEMO[1] & SEMOpx[2]. We collected and analyzed historical data and Transmission System Operator (TSO) predictions from 2019 to 2022, revealing that electricity prices exhibit significant volatility closely linked to demand and wind forecasts during this period, as illustrated in Fig. 1.

In the Irish DAM, prices are determined with hourly granularity, established at 11 pm on the preceding day. Bids for the DAM must be submitted before midday of the previous day to facilitate efficient market operations. Focusing on predicting DAM prices for the 24 settlement periods of the subsequent day. This temporal alignment ensures timely dissemination of forecasting insights, enabling market participants to strategise and make informed bidding decisions in advance. Our analysis incorporates a comprehensive set of regressors to predict DAM prices. These include historical DAM prices spanning the previous 168 h, alongside corresponding forecasts of demand and wind speed. Additionally, we integrate past 168-h DAM prices. This selection of attributes offers a robust foundation for price forecasting, capturing both historical trends and relevant external factors such as demand and weather conditions. For further details on our forecasting approach, market structure, datasets, and variables for both the DAM and BM, please refer to O'Connor et al. (2024a).

4 Methodology

This section delineates our study's methodology, focusing on key models and forecasting approaches across three primary stages: probabilistic forecasting

[1] https://www.sem-o.com/.
[2] https://www.semopx.com/market-data/.

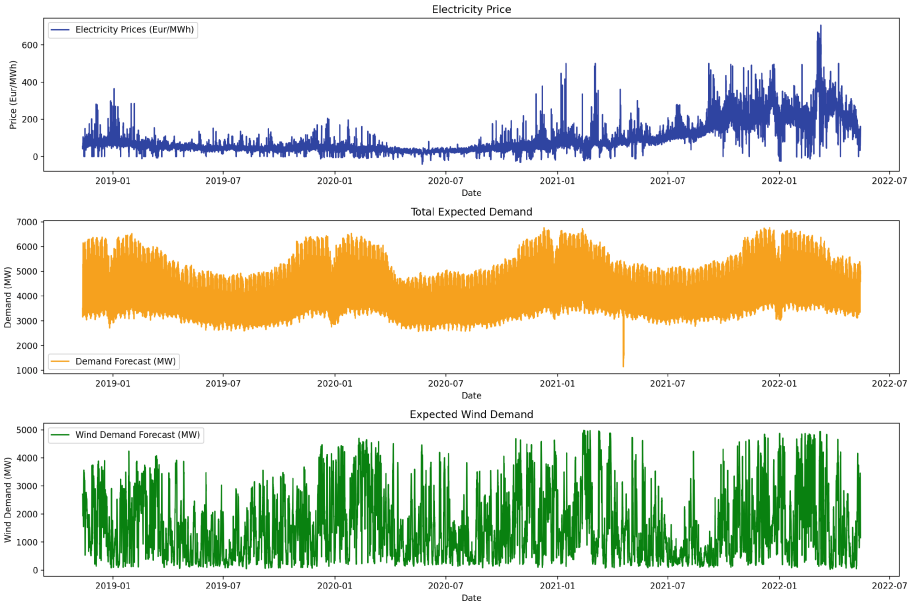


Fig. 1. Electricity Price and Demand Forecasts

models (Sect. 4.1), corresponding approaches (Sect. 4.2), and the evaluation of our models (Sect. 5.1). Our approach, encompassing both forecasting and trading, comprises five key steps:

1. *Data Collection and Preparation:* Aggregating historical and forward-looking data from the ISEM for the DAM.
2. *Data Pre-processing and Model Optimization:* We pre-process the data and optimize each predictive model. Hyperparameter tuning is conducted using three-month data subsets to enhance model performance.
3. *Walk-Forward Model Validation:* Performing iterative walk-forward validation to ensure the reliability of our models, continuously updating the time horizon.
4. *Quantile Forecasting:* Generating quantile forecasts using optimized models based on unseen test data, spanning 24 h.
5. *Financial Evaluation:* We compare all quantile pairs for each methodology in a single trade scenario to assess average forecasting model performance.

4.1 Quantile Regression Models

Our study employs quantile regression models, a statistical technique estimating conditional quantiles. These quantiles form a "quantile pair," defining a forecast range with lower and upper bounds, offering potential values within a specified confidence level. Specifically, we examine two quantile pairs: QP1, encompassing the 0.1–0.9 quantiles, denoted by $\alpha = 0.1$ and its complementary quantile 0.9,

and QP2, comprising the 0.3–0.7 quantiles. The probabilistic regressor models benchmarked are:

- *LASSO Estimated AR* (LEAR): A modified autoregressive time series approach incorporating LASSO regularization for improved performance and feature selection.
- *K-Nearest Neighbors* (KNN): A non-parametric instance-based learning approach that predicts an instance's output by comparing it to the "K" nearest neighbors in the training set.
- *Random Forest* (RF): An ensemble model that combines multiple regression trees.
- *Light Gradient Boosting Method* (LGBM): Similar to RF, LGBM uses multiple regression trees but follows the boosting principle.

4.2 Split Conformal Prediction

Split Conformal Prediction (SCP) (Shafer and Vovk (2008)) has limited application to time series data due to strict requirements for data exchangeability, it contributes valuable insights into constructing prediction intervals without relying on specific distribution assumptions.

Ensemble Batch Prediction Intervals (EnbPI). The EnbPI algorithm, as introduced in Xu and Xie (2021), emerges as a leading CP method tailored for dynamic time-series forecasting, effectively addressing the challenges posed by time series data without relying on data exchangeability assumptions. EnbPI's key features include its adaptability to dynamic time series, ensuring the importance of the sequence of data points is acknowledged, which is a crucial factor often overlooked by conventional CP methods. It provides PI with finite-sample, approximately valid marginal coverage, particularly for regression functions and time series with mildly mixing stochastic errors. EnbPI also demonstrates computational efficiency by avoiding overfitting without the need for data splitting or training multiple ensemble estimators.

Sequential Predictive Conformal Inference for Time Series (SPCI). The SPCI algorithm, introduced as an advancement on EnbPI in Xu and Xie (2023), presents a more versatile framework for time-series forecasting by directly leveraging the dependency of residuals when constructing PI, offering distinct advantages over previous methods. SPCI's key features include its utilization of residual dependencies, enhancing adaptability to dynamic time series. It employs a conditional quantile estimator rather than relying on empirical quantiles, resulting in more accurate PI estimates. SPCI also serves as a more general framework compared to both EnbPI and split conformal methods, capable of encompassing the functionalities of both through appropriate component selection. Furthermore, the computational efficiency of SPCI is maintained by fitting conditional quantile estimators using quantile models, ensuring effectiveness in sequential settings. For both EnbPI and SPCI, each model is run once, with bootstrap set to 15 for both the DAM and BM.

5 Experimental Results

This section analyzes probabilistic approaches and forecasting models in the DAM. Using diverse metrics, statistical tests, and financial indicators, we assess each model's efficacy in quantile regression and modern CP adaptations. Starting with Sect. 5.1, we analyze forecast accuracy through calibration, coverage, sharpness, and statistical testing metrics, aiming to evaluate the accuracy, reliability, and precision of probabilistic forecasts. Subsequently, in Sect. 5.2, we conduct a comparative analysis of outcomes across forecasting models, uncovering economic implications associated with each approach.

5.1 Evaluation

Our evaluation of probabilistic predictions focuses on two crucial dimensions: validity and efficiency. Efficiency, gauged through metrics like sharpness and interval width, enhances precision. Simultaneously, validity, including calibration and coverage, demands the generation of precise PI to affirm the reliability of our forecasts. Subsequent sections delve into each of these aspects with statistical testing. The evaluation culminates in the financial assessment of probabilistic approaches in Sect. 5.2. The Python code used for this section and the dataset are made available to ensure reproducibility GitHub[3].

Efficiency: Pinball Score and Interval Width. In the DAM, sharpness plays an important role in accurate anticipation, hedging, and real-time adjustments. The Pinball Score, derived from the Pinball Loss function, reflects the sharpness of the forecast based on the quantile prediction $\hat{q}_{\alpha,P}$ and observed price $P_{d,h}$: $PS(\hat{q}_{\alpha,P}, P_{d,h}, \alpha) =$

$$\begin{cases} (1-\alpha)(\hat{q}_{\alpha,P} - P_{d,h}) & \text{for } P_{d,h} \leq \hat{q}_{\alpha,p} \\ (\alpha)(P_{d,h} - \hat{q}_{\alpha,P}) & \text{for } P_{d,h} \geq \hat{q}_{\alpha,p} \end{cases} \tag{1}$$

Analyzing the Aggregate Pinball Score (APS) for DAM models, as illustrated in Table 1, with the lowest APS for marked in bold, we observe that both SCP and QR under-perform time-series adapted EnbPI and SPCI. Despite this underperformance, the top two models in APS are QR versions of RF and LGBM. This can be attributed to the high baseline accuracy of these models, limiting the potential improvement introduced by CP. In contrast, models with lower baseline accuracy, such as KNN and LEAR, experience a substantial enhancement with CP. LEAR sees a notable reduction in split CP and further in EnbPI in SPCI. Similarly, KNN exhibits a reduction from 6.65 to 5.71 for EnbPI and 6.09 for SPCI but faces a sharp increase in CP. This indicates that CP methods effectively mitigate the limitations of less accurate models, leading to significant improvements in forecast performance.

[3] https://github.com/ciaranoc123/PEPF_Conformal.

Table 1. Aggregate Pinball Score Scores. The best approach for each regressor is in bold.

Model	QR	CP	EnbPI	SPCI
KNN	6.65	7.79	**5.71**	6.09
LEAR	8.35	4.81	4.18	**4.08**
LGBM	**3.62**	3.67	3.76	3.71
RF	**3.63**	3.85	3.81	3.84
Avg.	5.56	4.99	**4.37**	4.43

Moving further into efficiency, the assessment extends beyond sharpness, focusing on width. While sharpness ensures reliability, efficiency, intricately linked to PI width, is pivotal in refining precision. Post-validity optimization enhances overall robustness and accuracy. An important highlight of the models utilizing CP is the interval width, showcased in Fig. 2. CP for highly accurate models reduces the interval width, while for less accurate models, it widens the interval width. This trend is evident in the consistent decline for EnbPI and SPCI models, where greater accuracy corresponds to a smaller interval and vice versa. This occurs due to the coverage guarantee, where less accurate models widen their intervals to meet this guarantee, a behaviour not observed in QR models. RF, while showing a similar high accuracy compared to LGBM, has a considerably different interval width, but it does achieve better coverage compared to other QR models. This holds for SCP, which, despite the high accuracy, fails to produce a narrow interval width as EnbPI and SPCI succeed with. The variance between the average Interval Widths is minimal.

Validity: Coverage and Kupiec Test. Both reliability and accuracy of probabilistic forecasts are vital, with a particular focus on evaluating their validity. For this, we examine the model's coverage, which entails assessing precision in capturing the price $P_{d,h}$ within predefined probability levels or intervals $(\hat{L}d, h^{\alpha}, \hat{U}d, h^{\alpha})$, with close scrutiny of the nominal coverage level α. Empirical coverage, indicating the alignment of predictions with specified intervals, is expressed through the binary indicator $I^{\alpha}_{d,h}$ for a given day d and hour h:

$$I^{\alpha}_{d,h} = \begin{cases} 1 \text{ for } P_{d,h} \in [\hat{L}^{\alpha}_{d,h} \hat{U}^{\alpha}_{d,h}] \\ 0 \text{ for } P_{d,h} \notin [\hat{L}^{\alpha}_{d,h} \hat{U}^{\alpha}_{d,h}] \end{cases} \tag{2}$$

Coverage metrics highlight the models' ability to capture the true distribution. Figure 3 provides a comprehensive coverage analysis across the 0.1–0.9 quantile range in the DAM, revealing the efficacy of diverse forecasting methodologies. In the DAM context, targeting a coverage of 0.8 for the 0.1–0.9 quantile range, CP models outperform QR counterparts. SCP, EnbPI, and SPCI models achieve commendable average coverages of 0.82, 0.83, and 0.80, respectively. In contrast, QR models lag significantly with an average coverage of 0.62, highlighting the

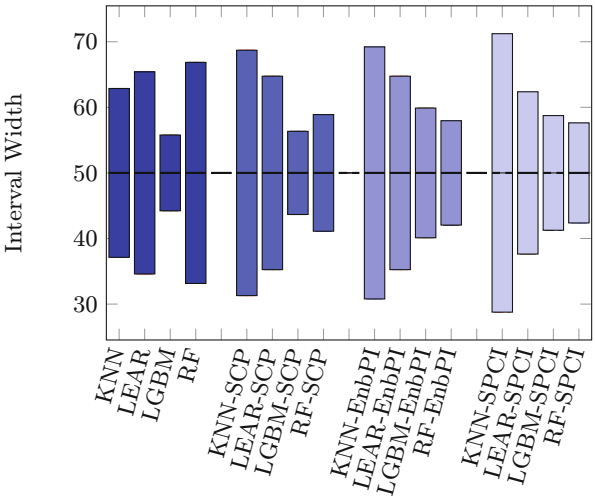


Fig. 2. Interval Width for each model in the DAM

superiority of CP methodologies in attaining target coverage levels and affirming their advantage over traditional QR techniques in probabilistic forecasting.

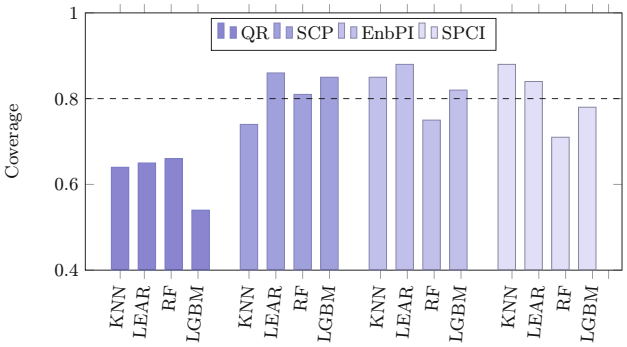


Fig. 3. Coverage for 0.1–0.9 quantile pair

In the 0.3–0.7 quantile pair analysis aiming for 0.4 coverage within the DAM framework (see Fig. 4), CP methodologies demonstrate pronounced impact. All CP-based models achieve the desired coverage, contrasting sharply with only one QR model meeting the criterion (RF being the sole exception). This stark contrast underscores CP's pivotal role in forecasting, ensuring robust coverage guarantees. The coverage performance gap between QR and CP approaches is significant. The QR model's average coverage is only 0.35, notably lower than that of CP-based strategies. SCP, EnbPI, and SPCI models yield average coverages of 0.59, 0.59, and 0.56, respectively, highlighting CP methodologies' superior

performance in attaining target coverage levels and enhancing the reliability of probabilistic forecasts within the dynamic electricity market domain.

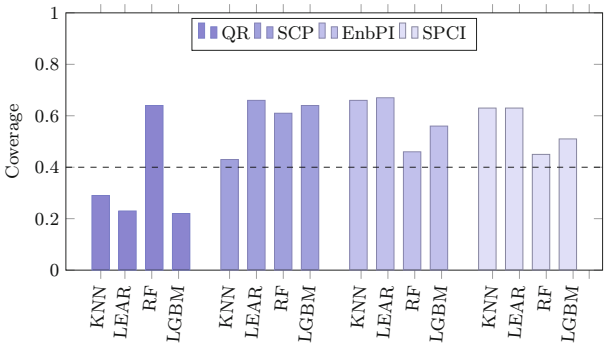


Fig. 4. Coverage for 0.3–0.7 quantile pair

The Kupiec test evaluates binary prediction model accuracy using labels (1 or 0) determined by predicted values falling within specified upper and lower bounds. For a model with T total observations, R events, predicted probability p, and significance level α, the test statistic is:

$$\chi^2 = -2\left[R\log(p) + (T-R)\log(1-p)\right]$$

The critical value χ^2_{crit} (one degree of freedom) is used to compare with X^2. If $\chi^2 > \chi^2_{crit}$, the null hypothesis of well-calibrated predictions is rejected.

To provide a comprehensive assessment, we perform the Kupiec test for unconditional coverage at 40% and 80% PI for each hour of the day, reporting the number of hours that pass the test.

Table 2. Number of Hours that pass the Kupiec test. QP1 = quantile pair 0.1–0.9, QP2 = quantile pair 0.3–0.7

Model	QR		SCP		EnbPI		SPCI	
	QP1	QP2	QP1	QP2	QP1	QP2	QP1	QP2
KNN	0	0	7	**20**	14	0	0	0
LEAR	0	0	3	0	0	0	9	0
RF	2	5	21	0	6	8	0	12
LGBM	0	0	7	0	22	0	**23**	0
Total	2	5	38	**20**	**42**	8	32	12

Table 2 provides insights into model performance across quantile pairs in the DAM, focusing on the hours passing the Kupiec Test, with the best results for

each QP marked in bold. Only the RF model among QR models passes the test, totalling 7 h out of 192. In contrast, CP approaches exhibit notable performance, exhibiting statistically significant performances 58 times. EnbPI models pass 50 h, while SPCI models pass 44 h. The significant difference in results between QR and CP approaches aligns with coverage outcomes in Figs. 3 and 4.

Efficiency and Validity: Winkler Score. The Winkler Score ($W_{t,h}$) amalgamates reliability and sharpness, providing a concise metric for the comprehensive assessment of probabilistic forecasts. It evaluates the width ($B_{t,h}$) of prediction intervals based on observed values ($y_{t,h}$), incorporating a penalty factor (α) for deviations from the interval bounds:

$$W_{t,h} = \begin{cases} B_{t,h} & \text{if } y_{t,h} \in [L_{t,h}, U_{t,h}] \\ B_{t,h} + \alpha(L_{t,h} - y_{t,h}) & \text{if } y_{t,h} < L_{t,h} \\ B_{t,h} + \alpha^2(y_{t,h} - U_{t,h}) & \text{if } y_{t,h} > U_{t,h} \end{cases}$$

Ultimately, Winkler Scores shed light on how well models ability to strike a balance between accuracy and interval width. Table 3 provides a detailed overview of Winkler Scores, offering insights into the efficiency and validity of different forecasting models in the dynamic electricity market landscape. The analysis in Table 3 highlight RF's dominance across all methodologies, with a Winkler Score of 22.89 in the QR framework. However, QR approaches are hindered by both KNN and LEAR's subpar results. In contrast, CP methods demonstrate consistent performance, with SPCI notably benefiting LEAR. Despite QR models, RF and LGBM, low accuracy and narrow interval width, challenges persist with models like KNN and LEAR. This underscores the need to explore alternative strategies, particularly CP approaches. Overall, accurate QR models excel in accuracy and precision in interval estimation, emphasizing their significance in probabilistic forecasting, especially in scenarios prioritizing accuracy and interval width.

Table 3. Winkler Scores. The best approach for each regressor is in bold.

Model	QR	CP	EnbPI	SPCI
KNN	57.02	64.16	**51.47**	56.16
LEAR	61.97	36.30	37.37	**35.06**
LGBM	**29.87**	30.39	32.76	31.31
RF	**22.89**	30.81	32.14	31.65
Avg.	42.94	39.51	**38.44**	38.55

Statistical Testing: Giacomini and White (2006) CPA Test. To draw statistically significant conclusions, we employ the Giacomini and White (Giacomini and White (2006)) Conditional Predictive Ability (CPA) test, utilizing a generalized Diebold-Mariano approach with a 24-dimensional vector of Pinball Scores for each day. The test statistic $\Delta_{X,Y,d}$ measures the difference between the L1 norms of APS vectors for models X and Y:

$$\Delta_{X,Y,d} = \|APS_{X,d}\| - \|APS_{Y,d}\|$$

where:

$$\|APS_{X,d}\| = \sum_{h=1}^{24} \sum_{\alpha=0.1}^{0.9} PS(\hat{q}_{\alpha,P}, P_{d,h}, \alpha)$$

for model X. P-values for the CPA test are computed for each model pair and dataset under the null hypothesis $H_0 : \phi = 0$ in the regression: $\Delta_{X,Y,d} = \phi' X_{d-1} + \epsilon_d$, where X_{d-1} contains day $d-1$ information, including a constant and lags of $\Delta_{X,Y,d}$.

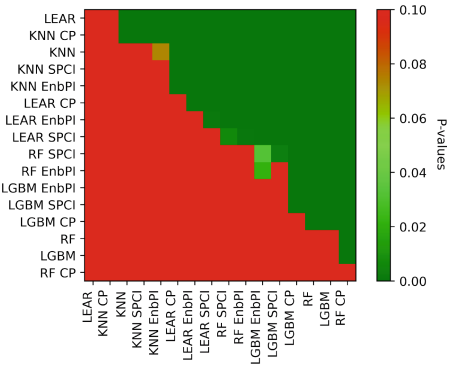


Fig. 5. Giacomini and White Test DAM

This test evaluates the reliability and precision of probabilistic forecasts across a 24 h horizon, offering insights into a model's capability to navigate evolving dynamics and uncertainties in electricity markets.

Figure 5 presents p-values using a chessboard representation. Dark green shades indicate the most significant differences between a model's forecast on the X-axis (better) and the forecast on the Y-axis (worse), with models arranged by APS. All CP methods, EnbPI, and SPCI, show statistically significant improvements for LEAR, suggesting enhanced forecasting accuracy. Conversely, KNN is significantly outperformed by all other models. RF, the top-performing model, demonstrates significant outperformance over all others, with SCP showing the best APS.

5.2 Financial Performance Analysis

This section outlines our BESS trading strategy, which is crucial for ensuring grid stability and seamlessly integrating renewable energy into the dynamic energy landscape. This is evaluated with our Single trade strategy (TS1). TS1 is a rule-based heuristic trading strategy adapted from Uniejewski and Weron (2021), Uniejewski (2023). This strategy utilizes quantile-based forecasts to optimize trading decisions involving a hypothetical 1 MWh battery with no discharge limit, 80% discharge efficiency, and 98% charge efficiency. Over the time horizon of 24 h, a single buy-sell pair trade is permitted, with the requirement that the buy trade occurs before the sell trade.

Table 4. Financial Performance Comparison of DAM Models. The best approach for each regressor is in bold.

Model	QR	CP	EnbPI	SPCI
KNN	€14,133	€14,342	**€14,488**	€13,374
LEAR	€2,889	€15,970	**€17,136**	€16,869
LGBM	€16,101	**€16,932**	€16,749	€16,676
RF	€16,652	**€16,883**	€16,862	€16,295
Avg.	€12,444	€16,032	**€16,309**	€15,804

In Table 4, CP adoption significantly improves LEAR and KNN models, with LEAR showing the most substantial enhancement. However, QR models' performance is notably impacted by LEAR and KNN, dragging down their average performance. Despite this, QR models perform comparably to CP approaches for LGBM and RF models, although CP models surpass QR for LGBM and two out of three for RF. Interestingly, although QR models like RF and LGBM exhibit superior APS and Winkler Scores, the inclusion of a coverage guarantee through CP appears to significantly influence financial outcomes. This highlights the nuanced interplay between model accuracy, interval width, and coverage assurance in financial performance evaluation. CP approaches demonstrate consistent performance across all models, highlighting the influence of CP's coverage guarantee in ensuring a robust forecasting framework compared to traditional quantile regression models.

5.3 Discussion

Model evaluation in the DAM reveals the intricate interplay between forecast accuracy, interval width, and financial outcomes. While QR models excel in accuracy and precision, challenges persist with models like KNN and LEAR, impacting financial metrics. In contrast, CP methodologies, including SCP, EnbPI, and SPCI, demonstrate reliability across all models, leveraging coverage guarantees for robust forecasting. Despite QR's accuracy, CP's coverage guarantee influences

financial outcomes, highlighting trade-offs between accuracy, interval width, and coverage assurance. QR and CP approaches complement each other, enhancing model performance and decision-making in the DAM.

6 Conclusion

In this study, we conducted a comprehensive analysis of probabilistic forecasting models within the dynamic electricity market, focusing on the DAM. Our investigation encompassed a diverse array of metrics, statistical tests, and financial indicators to evaluate the efficacy of traditional QR techniques and modern CP adaptations.

CP emerges as a potent approach for bolstering the reliability and precision of probabilistic forecasts within the DAM. CP's adaptive methodology addresses the intrinsic uncertainties of the DAM, offering superior coverage guarantees and interval width optimization compared to traditional QR techniques. SCP, EnbPI, and SPCI CP methodologies consistently outperform QR across various quantile pairs, exhibiting commendable performance in coverage metrics. CP effectively mitigates the limitations of less accurate models, resulting in substantial forecast improvements, making it a powerful tool for decision-making in the DAM. However, traditional QR models demonstrated exceptional accuracy and precision in interval estimation, underscoring their significance in probabilistic forecasting, despite challenges with certain models such as KNN and LEAR. Regarding financial performance, CP methodologies displayed remarkable consistency and reliability, leveraging their coverage guarantees to ensure robust forecasting frameworks.

Our analysis underscores the complementary roles of QR and CP approaches, with integrating CP alongside QR promising to advance probabilistic forecasting in DAM, facilitating more informed decision-making in dynamic electricity markets.

References

Abramova, E., Bunn, D.: Optimal daily trading of battery operations using arbitrage spreads. Energies **14**(16), 4931 (2021)

Dewolf, N., De Baets, B., Waegeman, W.: Valid prediction intervals for regression problems. Artif. Intell. Rev. **56**(1), 577–613 (2023)

EirGrid. Renewable energy (2022). https://www.eirgridgroup.com/how-the-grid-works/renewables/

Barber, R.F., Candes, E.J., Ramdas, A., Tibshirani, R.J.: Conformal prediction beyond exchangeability. arXiv e-prints, pages arXiv–2202 (2022)

Gammerman, A., Vovk, V., Vapnik, V.: Learning by transduction. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI) (1998)

Ghosh, S., Shi, Y., Belkhouja, T., Yan, Y., Doppa, J., Jones, B.: Probabilistically robust conformal prediction. In: Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI), vol. 216, pp. 681–690 (2023)

Giacomini, R., White, H.: Tests of conditional predictive ability. Econometrica **74**(6), 1545–1578 (2006)
Green, R., Vasilakos, N.: Market behaviour with large amounts of intermittent generation. Energy Policy **38**(7), 3211–3220 (2010)
Jianming, H., Luo, Q., Tang, J., Heng, J., Deng, Y.: Conformalized temporal convolutional quantile regression networks for wind power interval forecasting. Energy **248**, 123497 (2022)
Jensen, V., Bianchi, F.M., Anfinsen, S.N.: Ensemble conformalized quantile regression for probabilistic time series forecasting. IEEE Trans. Neural Netw. Learn. Syst. (2022)
Kath, C., Ziel, F.: Conformal prediction interval estimation and applications to day-ahead and intraday power markets. Int. J. Forecast. **37**(2), 777–799 (2021)
Khajeh, H., Laaksonen, H.: Applications of probabilistic forecasting in smart grids: a review. Appl. Sci. **12**(4), 1823 (2022)
Khosravi, A., Nahavandi, S.: Closure to the discussion of "prediction intervals for short-term wind farm generation forecasts" and "combined nonparametric prediction intervals for wind power generation" and the discussion of "combined nonparametric prediction intervals for wind power generation". IEEE Trans. Sustain. Energy **5**(3), 1022–1023 (2014)
Krishnamurthy, D., Uckun, C., Zhou, Z., Thimmapuram, P.R., Botterud, A.: Energy storage arbitrage under day-ahead and real-time price uncertainty. IEEE Trans. Power Syst. **33**(1), 84–93 (2017)
Narajewski, M.l., Ziel, F.: Optimal bidding on hourly and quarter-hourly day-ahead electricity price auctions: trading large volumes of power with market impact and transaction costs. arXiv preprint arXiv:2104.14204 (2021)
Lago, J., De Ridder, F., De Schutter, B.: Forecasting spot electricity prices: deep learning approaches and empirical comparison of traditional algorithms. Appl. Energy **221**, 386–405 (2018)
Lago, J., Marcjasz, G., De Schutter, B., Weron, R.: Forecasting day-ahead electricity prices: a review of state-of-the-art algorithms, best practices and an open-access benchmark. Appl. Energy **293**, 116983 (2021)
Leverger, C., et al.: Probabilistic forecasting of seasonal time series: combining clustering and classification for forecasting. In: International Conference on Time Series and Forecasting, pp. 47–63. Springer (2021)
Maciejowska, K., Nowotarski, J., Weron, R.: Probabilistic forecasting of electricity spot prices using factor quantile regression averaging. Int. J. Forecast. **32**(3), 957–965 (2016)
Marcjasz, G., Uniejewski, B., Weron, R.: Probabilistic electricity price forecasting with narx networks: Combine point or probabilistic forecasts? Int. J. Forecast. **36**(2), 466–479 (2020)
Marcjasz, G., Narajewski, M., Weron, R., Ziel, F.: Distributional neural networks for electricity price forecasting. arXiv preprint arXiv:2207.02832 (2022)
Martinez-Anido, C.B., Brinkman, G., Hodge, B.M.: The impact of wind power on electricity prices. Renew. Energy **94**, 474–487 (2016)
Nowotarski, J., Weron, R.: Recent advances in electricity price forecasting: a review of probabilistic forecasting. Renew. Sustain. Energy Rev. **81**, 1548–1568 (2018)
O'Connor, C., Collins, J., Prestwich, S., Visentin, A.: Electricity price forecasting in the irish balancing market. arXiv preprint arXiv:2402.06714 (2024a)
O'Connor, C., Collins, J., Prestwich, S., Visentin, A.: Optimizing quantile-based trading strategies in electricity arbitrage (2024b)

Oesterheld, C., Treutlein, J., Cooper, E., Hudson, R.: Incentivizing honest performative predictions with proper scoring rules. In: Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence, pp. 1564–1574 (2023)

O'Connor, C., Collins, J., Prestwich, S., Visentin, A.: Electricity price forecasting in the irish balancing market. Energy Strategy Rev. **54**, 101436 (2024). ISSN 2211-467X. https://doi.org/10.1016/j.esr.2024.101436. https://www.sciencedirect.com/science/article/pii/S2211467X24001433

Salem, T.S., Langseth, H., Ramampiaro, H.: Prediction intervals: Split normal mixture from quality-driven deep ensembles. In: Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), pp. 1179–1187 (2020)

Shafer, G., Vovk, V.: A tutorial on conformal prediction. J. Mach. Learn. Res. **9**(3) (2008)

Staffell, I., Rustomji, M.: Maximising the value of electricity storage. J. Energy Storage **8**, 212–225 (2016)

Tohidi, Y., Gibescu, M.: Stochastic optimisation for investment analysis of flow battery storage systems. IET Renew. Power Gener. **13**(4), 555–562 (2019)

Tzallas, P., Bezas, N., Moschos, I., Ioannidis, D., Tzovaras, D.: Probabilistic quantile multi-step forecasting of energy market prices: a UK case study. In: Artificial Intelligence Applications and Innovations, pp. 301–313. Springer (2022)

Uniejewski, B.: Smoothing quantile regression averaging: a new approach to probabilistic forecasting of electricity prices. arXiv preprint arXiv:2302.00411 (2023)

Uniejewski, B., Weron, R.: Regularized quantile regression averaging for probabilistic electricity price forecasting. Energy Econ. **95**, 105121 (2021)

Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World, vol. 29. Springer (2005)

Xu, C., Xie, Y.: Conformal prediction interval for dynamic time-series. In: International Conference on Machine Learning, pp. 11559–11569. PMLR (2021)

Xu, C., Xie, Y.: Sequential predictive conformal inference for time series. In: International Conference on Machine Learning, pp. 38707–38727. PMLR (2023)

Multivariate Human Activity Segmentation: Systematic Benchmark with ClaSP

Arik Ermshaus(✉), Patrick Schäfer, and Ulf Leser

Humboldt University of Berlin, Berlin, Germany
{ermshaua,patrick.schaefer,leser}@informatik.hu-berlin.de

Abstract. Human activity recognition (HAR) systems extract activities from observational data, such as sensor measurements from mobile devices, to provide for instance medical, fitness, or security information. A crucial initial step in these data analysis workflows is segmenting continuous numerical measurements into variable-sized segments that correspond to single activities. Human activity segmentation (HAS) enables downstream classification algorithms to label entire activities. Unfortunately, current time series segmentation (TSS) algorithms exhibit limited performance on multivariate sensor data due to complex temporal dynamics and irrelevant dimensions. This limits their applicability in HAR workflows. In this review, we provide a systematic benchmark of dimensionality reduction, model aggregation, and change point selection applied to the ClaSP TSS algorithm for real-world, multidimensional mobile sensing data. We evaluated the accuracy of the techniques in an experimental study using 250 data sets from the HAS challenge at ECML/PKDD and AALTD 2023. Our findings indicate that extending ClaSP for multivariate data, by aggregating internal representations, yields better results compared to reducing data dimensionality or selecting change points (CPs) from different channels. We report a new state of the art with 73% average accuracy on the challenge benchmark.

Keywords: Ubiquitous Sensing · Human Activity Recognition · Data Mining · Unsupervised Learning · Time Series Segmentation

1 Introduction

Monitoring human behaviour can provide crucial insights into personal health, fitness levels, and tactical arrangements of activities [1]. The sequence, duration, and classification of these activities are especially important as they are the basis for human processes. Health professionals, for instance, examine these processes to detect and monitor medical conditions [2], while military personnel analyse movement sequences to enhance decision-support systems [1]. To derive processes from monitoring human behaviour, data is acquired through videos,

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 18–34, 2025.
https://doi.org/10.1007/978-3-031-77066-1_2

Fig. 1. MTS from the HAS challenge benchmark [4]. The different channels show sensor signals capturing a student commuting to university, with activities colour-coded. Only a subset of dimensions is relevant for HAS. (Colour figure online)

environmental sensing, or wearable devices [3]. Smart devices are particularly valuable as they are worn consistently and can capture human behaviour using inertial measurement units (IMUs). These sensors typically include accelerometers, gyroscopes, and magnetometers, each producing triaxial measurements sampled at hundreds of Hertz (Hz). The resulting recordings form a multivariate time series (MTS), containing vectors of sensor measurements.

Figure 1 illustrates an example of a student commuting to university by train. The MTS captures human activities as extended periods of similar temporal patterns, such as waiting (orange) versus boarding (green). These patterns are repeated within processes and vary in shape and statistical properties across different activities. Note, acceleration (top) and magnetometer (bottom) readings are not consistent, which complicates knowledge discovery from such MTS.

To learn processes from human mobile sensing data, activities must be located and labelled. The extensive literature on human activity recognition (HAR) offers a comprehensive toolbox for feature extraction, classification, and software for this task [3]. One of the initial preprocessing steps in HAR involves segmenting MTS into smaller, consecutive parts, which are then classified using machine learning (ML) technology. This task can be approached by dividing the MTS into equal-sized subsequences or by segmenting it into variable-sized partitions, each corresponding to a single activity. The latter approach, known as human activity segmentation (HAS), is advantageous because it learns the start times and durations of activities, providing downstream tasks with clearly defined data segments corresponding to single activity labels [5]. HAS is generally referred to as time series segmentation (TSS), which divides a time series (TS) into homogeneous segments, separated by abrupt transitions called change points (CPs) [6].

Univariate TSS (UTSS) has been extensively studied and benchmarked [7,8], while multivariate TSS (MTSS) has received less attention. This is particularly problematic for activity segmentation, which requires multivariate signals to detect complex behaviours, such as dance or fight moves [9]. To address this shortcoming, the 8th Workshop on Advanced Analytics and Learning on Temporal Data (AALTD@ECML)[1] conducted the ECML/PKDD 2023 HAS discovery challenge [4]. The competition, featuring 57 participants, aimed to improve

[1] https://ecml-aaltd.github.io/aaltd2023.

the performance of multidimensional human activity segmentation using a new benchmark data set of 250 MTS capturing 100 activities performed by 15 students in 6 motion sequences. Both winners of the challenge utilized adaptations of the ClaSP method to apply it to multivariate measurements [10,11]. ClaSP has demonstrated superior performance for UTSS across multiple benchmarks for both batch [8] and streaming data [12]. To further explore its potential for offline MTSS, this benchmark reviews, categorizes, and evaluates currently available and new implementations of multivariate ClaSP to identify the most promising variant for multivariate HAS. Yet, not all studied methods are limited to ClaSP. Specifically, this paper's contributions are:

1. We review three different categories of approaches from the literature to modify the ClaSP method for offline MTSS, namely: dimensionality reduction, model aggregation and CP selection/ensembling. This includes technical descriptions, computational complexity analyses, and examples of HAR.
2. We conducted ablation studies for the three strategies and benchmarked their performances across the 250 HAS challenge data sets. Our findings show that model aggregation using distance averaging achieves the highest accuracy of 73%, setting a new state of the art.
3. To promote further development of MTSS algorithms, we provide all source codes, data sets, and extended evaluations on our supporting website [13].

The remainder of this paper contains background definitions (Sect. 2), related work (Sect. 3), the review and empirical evaluation of the three categories (Sect. 4 and 5), as well as a conclusion (Sect. 6).

2 Definitions and Background

We define the formal concepts of time series, subsequences, time series segmentation, and the ClaSP algorithm that we use throughout this paper.

Definition 1. *A time series (TS) T is an ordered sequence of $n \in \mathbb{N}$ real vectors $T = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n) \in \mathbb{R}^{n \times d}$ that constitutes the measurements from $d \in \mathbb{N}$ sensors of an activity routine of length n. The j-th sensor data is in $T^{(j)} \in \mathbb{R}^n$.*

If T has $d = 1$ dimension (aka channel), it is called a univariate TS (UTS); otherwise, it is a multivariate TS (MTS). The measurements within T are recorded at equidistant intervals, e.g., one data vector $\boldsymbol{t}_i$ every 20 ms.

In this way, we study MTS from IMU units in smartphones, which contain accelerometers, gyroscopes, and magnetometers [3]. Accelerometers capture the acceleration forces on a mobile phone, indicating motion presence. Gyroscopes measure angular velocity, inferring rotation during activities. Magnetometers record the geomagnetic field's impact on the smartphone, providing orientation information. IMU sensors capture measurements from the X, Y, and Z axes and are sampled at a few hundred Hertz (Hz), leading to long MTS that contain activities as recurring patterns. See Fig. 1 for an example.

Definition 2. *Given a TS T, a* subsequence $T_{s,e}$ *of T, with start and end offset s and e, is the d-dimensional window of contiguous measurements from T at position s to position e, i.e., $T_{s,e} = (\boldsymbol{t}_s, \ldots, \boldsymbol{t}_e)$, with $\boldsymbol{t}_i \in \mathbb{R}^d$ and $1 \leq s \leq e \leq n$. The width of $T_{s,e}$ is $|T_{s,e}| = e - s + 1$.*

Subsequences that capture single parts of an activity (e.g., taking a step) are called temporal patterns, and their length is the window size. TS with human behaviour contain parts with periodic subsequences constituting a longer activity (such as walking), which may suddenly change or gradually drift into another one (e.g., running). This manifests as changes in width, amplitude, or shape [14].

Definition 3. *For a TS T, capturing a motion routine, a* change point *(CP) is an offset $i \in [1, \ldots, n]$ corresponding to an abrupt transition between two activities. A* segmentation *of T is the ordered sequence of CPs in T, i.e., t_{i_1},...,t_{i_S} with $1 < i_1 < \cdots < i_S < n$ at which the captured behaviour changed activities.*

The task of human activity segmentation (HAS) is to find the segmentation of a TS that corresponds to the sequence of captured activities. Specifically, an algorithm must find all CPs that divide the TS into segments. This is an unsupervised machine learning (ML) problem generally referred to as time series segmentation (TSS) or change point detection (CPD) [6]. For MTS, it requires procedures to detect long stretches of homogeneous segments by selecting relevant channels and comparing subsequences by shape or statistical properties. A recent univariate TSS technique, which we study, is the ClaSP algorithm [8].

Definition 4. *Given an UTS T and a subsequence width w, ClaSP is a real-valued sequence of length $n - w + 1$, in which the i-th value marks the cross-validation score $c_i \in [0, 1]$ of a classifier trained on a binary classification problem with overlapping labelled subsequences $[(T_{1,w}, \boldsymbol{0}), \ldots, (T_{i-w+1,i}, \boldsymbol{0}), (T_{i-w+2,i+1}, \boldsymbol{1}), \ldots (T_{n-w+1,n}, \boldsymbol{1})]$, with labels 0 and 1.*

The central idea of ClaSP is to frame TSS as a collection of self-supervised, binary subsequence classification problems. Each cross-validation score c_i reports how well a TS classifier can differentiate the left from the right subsequences (see Fig. 5 for an example). All scores constitute a profile, that annotates T, show pronounced peaks for CPs and valleys during segments. This course is used for detecting CPs by finding the peaks [8].

The ClaSP concept has been efficiently implemented for batch [8] and streaming [12] settings using a k-nearest neighbour (k-NN) classifier. However, ClaSP originally has only been developed for UTS. To apply the idea also to MTS, several papers have proposed offline ensembling strategies to aggregate ClaSP profiles or the extracted CPs from multiple channels [10,11,15]. In this work, we review and systematically benchmark these proposals and new variants.

3 Related Work

In recent years, there has been much progress in human activity recognition (HAR) systems, devices, and experimental studies [1]. This research field is ver-

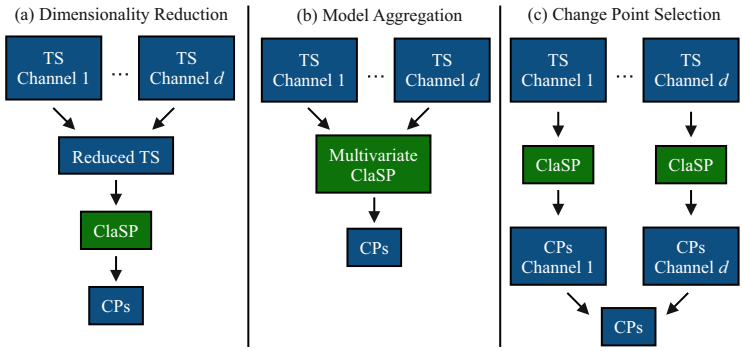


Fig. 2. Data flow for MTSS approaches. Blue boxes show input and output data. Green boxes display ClaSP component. (Colour figure online)

satile, encompassing various behaviours, data acquisition techniques, and analysis methods [3]. Some studies focus on the streaming setting, where activities are interpreted in real-time, providing users with immediate feedback, e.g. on their fitness status. Others analyse human behaviour post-hoc, deriving insights such as those from gait analysis [2].

Human activity segmentation (HAS) is a preprocessing step in HAR workflows. It partitions sensor data into variable-sized windows, each containing a single activity, which is then further processed [3]. The difficulty of this task is to select relevant sensor data for segmentation, ignore noise and misleading observations. Multivariate time series segmentation (MTSS) methods achieve this by dividing multidimensional numerical measurements into homogeneous slices based on shape or statistical properties [6]. A common approach is to frame this task as an optimization problem using a user-selected cost function, such as L2 or autoregressive cost, and applying a change point (CP) constraint. This problem can be solved with an exact algorithm like PELT or an approximation like BinSeg [6]. Specifically for multivariate sensor data from human activities, Sadri et al. proposed a cost function based on information gain [16]. ESPRESSO is another technique for multidimensional HAS that extracts CPs from a weighted subsequence arc curve and uses entropy to validate them [17]. The arc curve, proposed originally by Gharghabi et al., measures the density of similar subsequences in potential segments [7].

In this study, we explore the ClaSP algorithm in an offline multivariate setting for HAS. Initially proposed for UTS [8], ClaSP has demonstrated superior results on multiple HAS benchmarks compared to the aforementioned approaches [4, 5]. We review and evaluate several extensions of multivariate ClaSP to find a suitable implementation for HAR.

4 Multivariate Time Series Segmentation with ClaSP

In this section, we review three method-agnostic approaches for offline MTSS, we implemented with ClaSP: (a) dimensionality reduction, (b) model aggregation, and (c) CP selection/ensembling. The approaches aggregate multivariate into one-dimensional data at different stages of the segmentation process. Dimensionality reduction directly projects the multivariate input TS to an univariate one. An UTSS algorithm then processes this synthesized series to extract CPs. Model aggregation extends internal data structures for multivariate input, merging them into univariate aggregates for segmentation. CP selection/ensembling computes segmentations separately for each TS dimension, then filters out or merges (near-)duplicate CPs.

Figure 2 shows the data flow of the approaches. Dimensionality reduction and CP selection/ensembling operate directly on the input or output data (blue boxes) and are method-independent. Model aggregation modifies ClaSP's internal components (green box) and requires changes to the original algorithm.

In the following Subsects. 4.1 to 4.3, we review the three strategies (a to c) in detail. We provide own pseudocodes, analyse their implementations and computational complexities, as well as discuss advantages and shortcomings.

4.1 Dimensionality Reduction

Univariate TS analytics can be applied to MTS by reducing their dimensionality. Tanaka et al. [18] explored this approach in the context of motif discovery. The main idea is to treat MTS channels as features and time points as instances. Then, a dimensionality reduction technique aggregates these features into a single component. Dimensionality reduction methods project multidimensional features of a data set into a lower-dimensional target space while preserving important relationships between samples as good as possible, such as distributions, separability, or distances [19]. This is achieved by merging correlated features, identifying independent components, or projecting features. Popular techniques include principal component analysis (PCA), independent component analysis (ICA), random projection (RP), and autoencoders.

Algorithm 1 implements this idea. It takes a MTS T, $|T| = n$ with d dimensions as input and applies a dimensionality reduction technique, such as PCA [19], to reduce the d channels into one. UTSS algorithms, such as ClaSP, can then process this synthesized TS. Reducing TS dimensionality decreases noise and redundant information from data. However, it can also over-simplify complex temporal dynamics or destroy inter-channel dependencies.

The computational complexity of Algorithm 1 mainly depends on the dimensionality reduction method used [19]. For instance, PCA requires $\mathcal{O}(n \cdot d^2 + d^3)$ for calculating the covariance matrix and singular value decomposition (SVD). ICA needs $\mathcal{O}(n \cdot d^2 \cdot i)$ for i iterations, and random projection takes $\mathcal{O}(n \cdot d \cdot l) = \mathcal{O}(n \cdot d)$ for multiplying X with a random projection matrix of size $(d \times l)$, where $l = 1$ is the size of the lower-dimensional target space. Regarding space complexity, PCA

Algorithm 1. Dimensionality Reduction

```
procedure ReduceDimensions(T)
    X ← create data set matrix from T of size (n × d)
    R ← PCA(n_components = 1).fit_transform(X)  ▷ Apply reduction technique.
    return R
end procedure
```

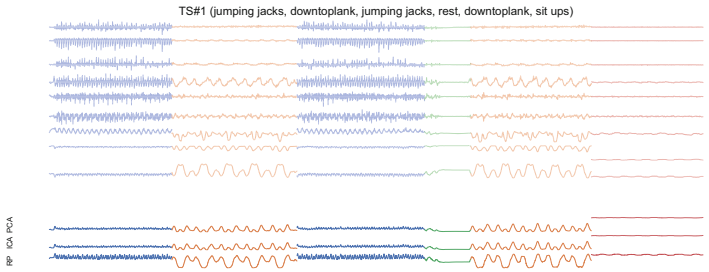


Fig. 3. TS dimensionality reduction with PCA, ICA and RP. The top-9 channels show sensor signals capturing indoor sport activities [4] (different light colours). The bottom-3 TS illustrate the results of PCA, ICA and RP (full colours). (Colour figure online)

and ICA each require $\mathcal{O}(n \cdot d + d^2)$ to store X and the covariance (or unmixing) matrix. Random projection requires $\mathcal{O}(n \cdot d + d \cdot l)$, which reduces to $\mathcal{O}(n \cdot d)$.

Figure 3 illustrates an example of dimensionality reduction. It shows an indoor sports routine performed by a 25-year-old male, captured by three triaxial inertial measurement unit (IMU) smartphone sensors as a MTS (top-9 channels). Activities are coloured differently. The bottom-3 UTS display the synthesized TS using PCA, ICA, and RP. Each technique uniquely reduces the MTS but retains similarities within activities (e.g., two instances of "down to plank") and separability across activities (e.g., "jumping jacks" versus "sit-ups"). This indicates that dimensionality reduction can be a viable preprocessing for MTSS.

4.2 Model Aggregation

Instead of directly reducing a MTS to an UTS, model aggregation adapts the TSS procedure for multivariate data. The central idea is to compute the TSS model per channel and aggregate the resulting features, allowing for potentially more meaningful representations compared to raw measurements. Model aggregation is well suited for the ClaSP algorithm that involves a two-step feature transformation, k-NN model, and cross-validation score profile, both of which can be extended for MTS and aggregated.

Distance Averaging: To create the k-NN model for MTS, we calculate distances between subsequences for each channel separately using STOMP, which employs optimization techniques to speed-up calculations [20]. Then, we average

Algorithm 2. Distance Averaging

procedure AVERAGEDISTANCES(T, w, p)
  $D_{global} \leftarrow$ array of length $|T| - w + 1$ with 0s ▷ Initialize distances.
  for $i \in [1, \ldots, d]$ **do** ▷ Update distances for each dimension.
    $D_{global} \leftarrow D_{global} +$ CALC_DISTANCES($T^{(i)}, w, p$)
  end for
  return $\frac{1}{d} \cdot D_{global}$ ▷ Return averaged distances.
end procedure

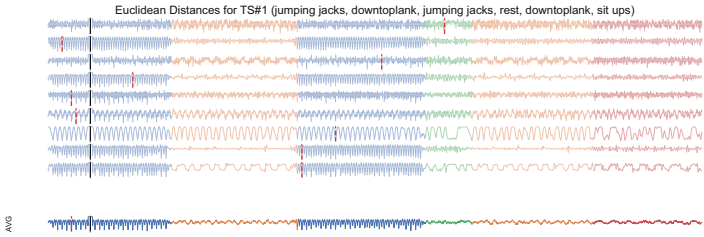


Fig. 4. Euclidean distances between the subsequence at position 500 (black bar) and all other subsequences (width 50) for the TS from Fig. 3 (light colours). 1-NNs are illustrated as red bars. Ideally, they should be located in one of the blue segments. The bottom distances are the averages (full colours). (Colour figure online)

these distances and thereafter select the k smallest values, applying an exclusion radius to ignore self-matches. For constructing the k-NN in ClaSP, we calculate the z-normalized Euclidean distances between subsequences of width w.

Algorithm 2 outlines this process. It takes a MTS T, $|T| = n$ with d channels, a subsequence width w, and a position p as input and averages the pairwise distances between $T^{(i)}_{p,p+w-1}$ and all other subsequences in $T^{(i)}$ for each of the d dimensions. This includes proximity information from each dimension, weighing them equally. It captures complex multivariate dynamics, but is sensitive to noise or irrelevant channels. Thereafter, the k smallest values are determined.

The runtime complexity of Algorithm 2 is mainly driven by the distance calculation, which is $\mathcal{O}(n)$ for a single channel using STOMP [20]. As it is executed d times, the overall runtime is $\mathcal{O}(n \cdot d)$, increasing the complexity of ClaSP from $\mathcal{O}(n^2)$ for the univariate case to $\mathcal{O}(d \cdot n^2)$ for MTS with d dimensions. The space complexity for distance averaging, and for ClaSP, is $\mathcal{O}(d \cdot n)$.

Figure 4 demonstrates Algorithm 2 for the TS from Fig. 3 and the subsequence at position 500 (black bar, width 50). The top-9 series show the distances for each channel, and the bottom series shows the averaged distances, the output of the procedure. The red bars indicate the 1-NN subsequence per channel. The first channel wrongly assigns a subsequence from the resting activity, which could lead to an inaccurate segmentation. All other dimensions correctly find a subsequence from the first and second instance of jumping jacks. The averaged distances also identify the 1-NN belonging to the first segment.

Algorithm 3. Profile Averaging

```
1: procedure AverageProfiles(T, w)
2:     P_global ← array of length |T| − w + 1 with 0s          ▷ Initialize profile.
3:     for i ∈ [1, ..., d] do                                  ▷ Update scores for each dimension.
4:         P_global ← P_global + calc_clasp(T^(i), w)
5:     end for
6:     return (1/d) · P_global                                 ▷ Return averaged profile.
7: end procedure
```

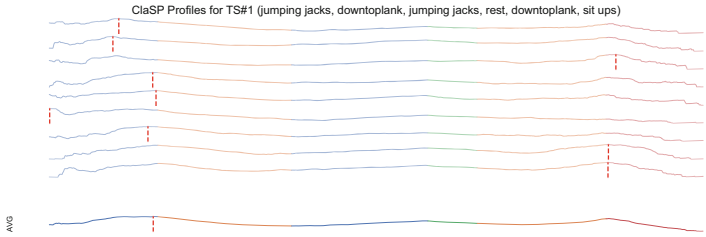


Fig. 5. ClaSP profiles for the TS from Fig. 3 (light colours). Global maxima (CPs) are illustrated as red bars. Ideally, they should capture one of the activity transitions. The bottom profile is the average (full colours). (Colour figure online)

Distance-Based Channel Selection: Distance averaging assumes all TS channels to be equally important, which may not be true in practice. Often, only a fraction of the dimensions contain relevant information for segmentation (e.g., top 2–9 distances in Fig. 4). Considering all channels in such scenarios introduces unnecessary noise into the distance calculations. To address this, we can modify Algorithm 2 to select distances based on specific criteria.

One simple approach is to consider only the f out of d distance vectors with increasing global minima. This subgroup has the smallest average 1-NN distance. The parameter f can be user-defined or learned through thresholding. A more sophisticated version of this technique is implemented in the mSTAMP algorithm [21] for motif discovery. It calculates distances for all dimensions, sorts them in ascending order per time point, and averages the first f out of d distance vectors, fitting best for segmentation. This subgroup not only includes the smallest 1-NN distances, but also the smallest overall distances at each offset. Yet, the f dimensions are not fixed, but can change.

Profile Averaging: Score profiles annotate the likelihood of CPs for each time point. They can be applied to each channel of a MTS, with the resulting profiles being averaged before applying segmentation. This concept was discussed by Wang et al. [15] for ClaSP and is also implemented for FLUSS [7].

Algorithm 3 implements this idea. It takes a MTS T, $|T| = n$ with d channels and a subsequence width w as input, computes ClaSP for each of the d channels and averages the resulting profiles, which is used for segmentation. Similar to distance averaging, it considers each profile as an equal contribution. How-

Algorithm 4. Change Point Selection/Ensembling

```
procedure FilterChangePoints(T)
    C_all ← empty set                                  ▷ Initialize CP set.
    for i ∈ [1, ..., d] do                             ▷ Add CPs for each dimension.
        C_all ← C_all ∪ clasp_segmentation(T^(i))
    end for
    return filter(C_all)                               ▷ Filter/merge (near-)duplicate CPs.
end procedure
```

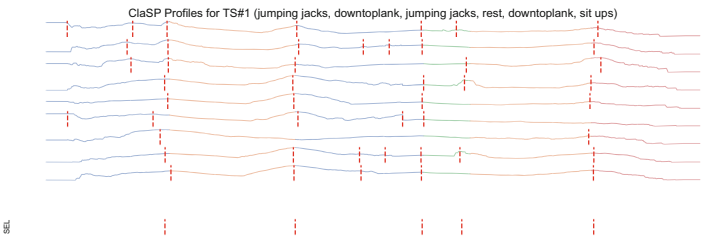


Fig. 6. ClaSP profiles for the TS from Fig. 3 (light colours). All found CPs are illustrated as red bars. Single channels contain false or missing predictions. The bottom CPs are the filtered selection and correctly capture the activity transitions. (Colour figure online)

ever, it assumes that individual channels contain sufficient differences to identify segment transitions, recognized by the k-NN classifier and represented as peaks in the profiles. In contrast, distance averaging detects differences spread across multiple dimensions, recognized only after combining distances.

The computational complexity of Algorithm 3 depends on the ClaSP routine, which runs in $\mathcal{O}(n^2)$ per UTS. This process repeats for each channel, resulting in a total runtime complexity of $\mathcal{O}(d \cdot n^2)$. The space complexity for a single ClaSP is $\mathcal{O}(n)$, which extends linearly to $\mathcal{O}(d \cdot n)$ in the multivariate setting.

Figure 5 illustrates how Algorithm 3 computes ClaSP per channel (top-9 profiles) and then averages them (bottom profile). The red bars indicate the highest-scoring CPs per channel (global maxima), mainly located at the first transition (jumping jacks to plank) and the last change (plank to sit-ups). However, the CPs from the first two (and fifth) dimensions substantially deviate from the ground truth. The averaged profile correctly aggregates the CP information, identifying the first transition as the most substantial CP.

Score-Based Channel Selection: Similar to distance averaging, Algorithm 3 assigns equal weight to each channel during aggregation, which may wrongly include noisy or misleading scores (see profiles 1 & 2 in Fig. 5). To address this, we can apply different selection strategies to filter dimensions.

ClaSP uses significance testing to validate CPs in segmentation. We can apply this testing before averaging to include only profiles with significant CPs. However, this may exclude channels showing trends in their ClaSP that might

become significant when combined. Another selection technique, used in ClaSP ensembling, is to maximize scores rather than averaging them. This approach selects the best-performing ClaSP by design and disregards all but one channel in MTS aggregation, potentially filtering out useful channels. We can counteract this by selecting the top f out of d best-scoring profiles.

4.3 Change Point Selection/Ensembling

Instead of modifying a MTS or the segmentation model, we can perform TSS on each channel separately and only merge the resulting CPs. This technique is invariant to the segmentation procedure and primarily relies on CP filtering, akin to dimensionality reduction. The authors of [11] propose an interval weighting scheme, while Harańczyk uses threshold-based pruning [10]. Another approach is to use agglomerative CP clustering.

Algorithm 4 details CP selection. It takes a MTS T, $|T| = n$ with d channels as input, computes the CPs for all d dimensions separately and filters them, e.g., using threshold-based pruning [10], to remove near-duplicates and return unique CPs. Similar to profile averaging, this procedure can only identify CPs recognized per channel, not those requiring the combination of dimensions.

The runtime complexity of Algorithm 4 primarily depends on ClaSP and the specific filtering mechanism. Running the univariate ClaSP segmentation per channel costs $\mathcal{O}(C \cdot n^2)$ for C detected CPs. Extending this to the multivariate setting results in $\mathcal{O}(d \cdot C_{max} \cdot n^2)$ for a maximum of C_{max} detected CPs per channel. Applying the filtering mechanism adds $\mathcal{O}(d \cdot C_{max})$ for interval weighting, $\mathcal{O}(C_{all}^2)$ for threshold-based pruning, and $\mathcal{O}(C_{all}^3)$ for agglomerative clustering. The space complexity of the entire Algorithm 4 is $\mathcal{O}(d \cdot n)$, except for agglomerative clustering, which adds $\mathcal{O}(C_{all}^2)$ for storing the CP distance matrix.

Figure 6 shows Algorithm 4 for the MTS from Fig. 3. The top-9 series are ClaSP profiles with all detected CPs per channel, and the bottom CPs are filtered using agglomerative clustering, reporting the mean CP per cluster. While individual channels produce comparable but heterogeneous segmentations with a mix of true, false, and missing CP predictions, the filtered output correctly captures the true activity transitions. Thus, it can be favourable to ensemble CPs from multiple channels, compared to selecting predictions from one dimension.

5 Experimental Evaluation

We empirically investigate the accuracy of the three strategies for MTSS using human activity data. First, we describe the used mobile sensing data sets and evaluation measure in Subsect. 5.1. We then explore design choices per approach in an ablation study (see Subsect. 5.2). Lastly, we compare the performance of the techniques in Subsect. 5.3 with 3 univariate baselines and explore the segmentation of an use case. All experiments were conducted on an Intel Xeon 8358 with 2.60 GHz, 2 TB RAM, 128 cores, running Python 3.8. To foster reproducibility and replicability of our results, we provide source codes, data sets, and raw measurements on our supporting website [13].

5.1 Setup

HAS Challenge Data Sets: We utilize the 250 MTS from the Human Activity Segmentation Challenge [4], held at ECML/PKDD and the AALTD 2023 workshop. This multi-modal benchmark originates from 40 twelve-dimensional smartphone recordings, capturing 15 students performing 6 distinct motion sequences in both indoor and outdoor settings. Each data set includes values from 6–9 out of 12 sensors: triaxial accelerometer, gyroscope, magnetometer as well as latitude, longitude, and speed. The MTS are synchronized and sampled at 50 Hz, annotated with activity labels and transition offsets as ground truth. They range from 7 s to 14 min in duration (median: 100 s) and contain between 1 and 15 segments (76% of MTS have 5 or fewer). The duration of single activities varies from half a second (waiting) to 10 min (running). The 250 MTS come in fixed randomly split public and private sets (125 MTS each) to avoid overfitting; the public set is used for design choices, and the private set is used for competitor comparisons. For an example TS, see Fig. 1 or 3 again.

Covering Score: We use the Covering evaluation measure [5], to assess the performance of MTSS methods on a given HAS data set. It measures how well the predicted segments overlap with the annotated ones using the Jaccard index.

Specifically, let T, $|T| = n$ be a MTS with ground truth CPs $cpts_{true}$ and predicted CPs $cpts_{pred}$, each located in $[1, \ldots, n]$. We consider the interval of successive CPs $[t_{i_k}, \ldots, t_{i_{k+1}}]$ a segment in T, and let $segs_{true}$ and $segs_{pred}$ be the sets of ground truth and predicted segmentations, respectively. We define $t_{i_0} = 0$ as the first and $t_{i_C} = n + 1$ as the last CP to include the first (last) segment. Covering reports the best-scoring weighted overlap between the true and predicted segmentations as a value in the interval $[0, \ldots, 1]$ (higher is better):

$$Covering = \frac{1}{\|T\|} \sum_{s \in segs_{true}} \|s\| \cdot \max_{s' \in segs_{pred}} \frac{\|s \cap s'\|}{\|s \cup s'\|} \tag{1}$$

This allows for the comparison of sets $cpts_{true}$ and $cpts_{pred}$ with varying lengths, including empty sets. To compare different methods across multiple data sets, we aggregate the Covering scores into a single ranking. We compute the rank of each technique per data set, i.e., the best method is assigned rank 1, the second-best rank 2, etc., and average the ranks over all data sets per method to obtain its overall rank. For illustration, we use critical difference (CD) diagrams [22]. The best approaches, with the lowest average ranks, are shown to the right of the diagram (see Fig. 7 left). Groups of methods with insignificantly different performances are connected by a bar, based on a pairwise one-sided Wilcoxon signed-rank test with $\alpha = 0.05$ and Holm correction.

Hyperparameters: We set different hyperparameters for the approaches and fix them for all evaluations. The dimensionality reduction techniques reduce a MTS to a lower-dimensional target space. We use the first component for all

tested methods and draw random values from a normal distribution for random projection. ClaSP processes the synthesized TS with default parameters.

The selection strategies for model aggregation require the number of selected dimensions, which we set to $\lfloor \frac{d}{2} \rfloor$ for all techniques that require this parameter. As the choice of this parameter is generally use case dependent, we study the overall impact of excluding dimensions in the segmentation. The multivariate ClaSP, used for model aggregation, first learns a window size per channel and then uses the minimal value for remaining computations. It also sets a stricter significance level for CP validation of $1e-30$ (compared to univariate ClaSP's default of $1e-15$), which overall leads to more accurate results.

For CP ensembling, ClaSP processes each channel with default parameters. Agglomerative clustering uses average linkage thresholded at $5 \cdot w$. Clusters of CPs at or above this distance are not merged, following ClaSP's strategy that segments must have at least $5 \cdot w$ data points. Pruning and weighting-based CP detection make more custom design choices, as outlined in the respective publications [10,11]. We evaluate the winning challenge submissions.

Additionally, we report the scores for the univariate BinSeg, ClaSP, and FLUSS challenge baselines, which only segment Y-axis acceleration [4].

5.2 Ablation Study

We tested different design choices for each of the three approaches of MTSS with ClaSP on the public HAS challenge data split (125 TS). We analyse the performances per approach to identify variants for comparative evaluation. CD diagrams and more detailed analyses are on our supporting website [13].

Dimensionality Reduction: We tested PCA, ICA, and RP for dimensionality reduction. PCA (1.93) ranked first, followed by RP (1.96) and ICA (2.11). The differences in rank are not statistically significant. Considering average Covering, RP ($67 \pm 24\%$) leads, followed by PCA ($64 \pm 26\%$) and ICA ($63 \pm 27\%$). We find that the specific implementation only leads to differences in tendencies. Both RP and PCA are good candidates for this approach. We choose PCA for further analysis as it scores the lowest average rank.

Model Aggregation: We evaluated distance averaging with all dimensions, using the f out of d distances with the smallest minima, and dimension sorting for selection. Using all channels ranks first (1.93), followed by using f out of d distances (1.99), and dimension sorting (2.08). Differences in rank are not significant. However, using all distances also has the highest Covering of $74 \pm 21\%$ on average and 78% in median, compared to the selection strategies. We conclude that distance-based channel selection does not have an advantage for the TS.

For profile averaging, we tested using all dimensions, aggregating f out of d profiles with the highest maxima, and averaging profiles with significant CPs. Using f out of d profiles ranks first (1.88), averaging all profiles ranks second (1.9), and aggregating profiles with significant CPs ranks last (2.22). While the two best-scoring variants do not show significant differences in rank, both are significantly better than the last. Summary statistics confirm the ranking (see [13]).

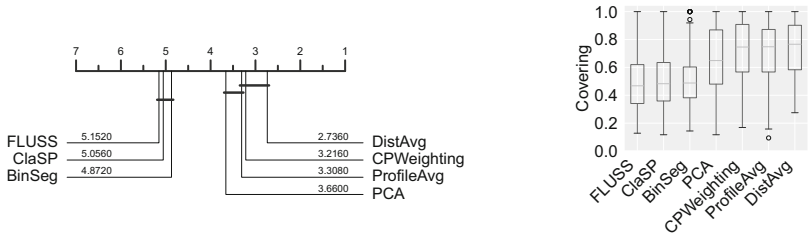


Fig. 7. CD diagram (left) and box plot (right) on the private HAS challenge split for the univariate challenge baselines (BinSeg, ClaSP, FLUSS) as well as dimensionality reduction with PCA, distance/profile averaging, and CP weighting.

Similar to distance averaging, the selection strategies do not show any substantial improvement in scores or ranks. On the contrary, averaging profiles with significant CPs even leads to significantly worse results.

We find that for both distance and profile averaging, it could be the case that (a) most dimensions positively contribute to the segmentation, (b) the selection criteria do not work as well, or (c) the hyper-parameter setting the number of selected dimensions needs more careful adjustment. Based on our results, we use all channels in model aggregation for further comparison.

CP Selection/Ensembling: We assessed CP selection and ensembling using pruning, weighting, and agglomerative clustering. Weighting (1.95) ranks first, followed by pruning (2.02) and clustering (2.03). Differences in rank are not significant. Summary statistics confirm the ranking with average Covering between 69–72% and standard deviations of 21–25%. Here again, we find that the specific implementation of CP selection/ensembling accounts for only small differences in performance. We choose the best-ranking approach, CP weighting.

5.3 Comparative Evaluation

We evaluated the selected variants from three MTSS approaches and the univariate challenge baselines on the private HAS challenge data split (125 TS) to determine the best performance. Figure 7 (left) displays the mean ranks on Covering score. Distance averaging (2.74) achieves the best results, followed by CP weighting (3.22), profile averaging (3.31), PCA (3.66), univariate BinSeg (4.87), ClaSP (5.06), and FLUSS (5.15). The differences in rank for the top-3 approaches are not significant. However, all multivariate variants significantly outperform the univariate baselines. Distance averaging wins (or ties) for 53 TS, followed by profile averaging (38), CP weighting (35), PCA (33), BinSeg/ClaSP (12), and FLUSS (11). The counts do not add up to 125 due to ties. The multivariate variants only insignificantly outperform the univariate baselines for data sets with one segment (16 instances) or two segments (13). For three or more segments (96), the results align with the global ranking in Fig. 7 (left).

Fig. 8. Top-2: X-axis acceleration and magnetometer from Fig. 1 (activities are differently coloured). Bottom-3: ClaSP profiles created using PCA, distance and profile averaging. CPs are illustrated as red bars. Distance averaging shows best results. (Colour figure online)

Regarding summary statistics (Fig. 7 right), distance averaging scores the highest Covering of $73 \pm 20\%$ in mean and 77% in median, with minimal differences to the 2nd and 3rd place, but a huge advance of at least 21% points in mean to the challenge baselines. Similarly, distance averaging outperforms the other approaches in pairwise comparisons in at least 52% of cases.

As an example, Fig. 8 illustrates the ClaSP profiles for PCA, distance averaging, and profile averaging (bottom-3) for the MTS from Fig. 1. Extracted CPs are shown as red bars. Distance averaging shows peaks, captures all transitions with CPs and introduces a false positive during the train ride, possibly due to misleading magnetometer readings showing deflections throughout this activity. PCA and profile averaging result in shallow profiles with missing CPs or more false positives. This demonstrates that distance averaging is interpretable for human inspection and has the best performance.

In conclusion, we find that distance averaging is the most promising approach for offline MTSS with ClaSP on the challenge data. It outperforms, yet not significantly, CP weighting and profile averaging on the private split and other model aggregation methods on the public split. It also achieves the highest average Covering score and sets a new first place on the challenge data (private split) with an F1-score of 52%, compared to CP selection with pruning at 51.5% (according to [4]). Overall, this evaluation suggests that aggregating intermediate representations of ClaSP is more effective than reducing MTS or CPs directly. However, the performance results indicate room for improvement.

6 Conclusion

In this paper, we studied and evaluated three categories to extend the ClaSP method to the multivariate setting. Dimensionality reduction and CP selection/ensembling have the advantage of being independent of the ClaSP method and can be used to extend other UTSS algorithms to the multidimensional setting. Model aggregation using distance averaging is specific to ClaSP, but leads to the best-scoring and interpretable results, setting a new state of the art on the HAS challenge data sets. Hence, we recommend using this ClaSP extension when applying the method to multivariate sensor data.

However, the performance improvement of distance averaging is small and statistically insignificant compared to CP ensembling using weighting and profile averaging. Future work should empirically investigate autoencoders for dimensionality reduction, the scalability and runtime of MTSS procedures, and see if they transfer to other domains, such as medical condition monitoring or IoT.

References

1. Lara, O.D., Labrador, M.A.: A survey on human activity recognition using wearable sensors. IEEE Commun. Surv. Tutorials **15**, 1192–1209 (2013)
2. Zhou, L., Fischer, E., Brahms, C.M., Granacher, U., Arnrich, B.: Duo-gait: a gait dataset for walking under dual-task and fatigue conditions with inertial measurement units. Sci. Data **10**(1), 543 (2023)
3. Ahad, M.A.R., Antar, A.D., Ahmed, M.: Iot sensor-based activity recognition - human activity recognition. Intell. Syst. Ref. Libr. (2021)
4. Ermshaus, A., et al.: Human activity segmentation challenge @ ECML/PKDD'23. In: AALTD@ECML/PKDD (2023)
5. Ermshaus, A., Singh, S., Leser, U.: Time series segmentation applied to a new data set for mobile sensing of human activities. In: EDBT/ICDT Workshops (2023)
6. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. Signal Process. **167**, 107299 (2020)
7. Gharghabi, S., et al.: Domain agnostic online semantic segmentation for multi-dimensional time series. DMKD **33**, 96–130 (2018)
8. Ermshaus, A., Schäfer, P., Leser, U.: ClaSP: parameter-free time series segmentation. DMKD **37**, 1262–1300 (2023)
9. Matsuyama, H., Hiroi, K., Kaji, K., Yonezawa, T., Kawaguchi, N.: Ballroom dance step type recognition by random forest using video and wearable sensor. In" UbiComp/ISWC, pp. 774–780 (2019)
10. Harańczyk, G.: Change points detection in multivariate signal applied to human activity segmentation. In: AALTD@ECML/PKDD (2023)
11. Huang, T.-J., Zhou, Q.-L., Ye, H.-J., Zhan, D.-C.: Change point detection via synthetic signals. In: AALTD@ECML/PKDD (2023)
12. Ermshaus, A., Schäfer, P., Leser, U.: Raising the ClaSS of streaming time series segmentation. PVLDB **17**(8), 1953–1966 (2024)
13. Multivariate ClaSP Code, Extended Experiments and Raw Results (2024). https://github.com/ermshaua/multivariate-clasp
14. Ermshaus, A., Schäfer, P., Leser, U.: Window size selection in unsupervised time series analytics: a review and benchmark. In: AALTD@ECML/PKDD (2022)
15. Wang, C., Wu, K., Zhou, T., Cai, Z.: Time2State: an unsupervised framework for inferring the latent states in time series data. PACMMOD **1**(1), 1–18 (2023)
16. Sadri, A., Ren, Y., Salim, F.D.: Information gain-based metric for recognizing transitions in human activities. PMC **38**, 92–109 (2017)
17. Deldari, S., Smith, D.V., Sadri, A., Salim, F.D.: ESPRESSO: entropy and shape aware time-series segmentation for processing heterogeneous sensor data. In: IMWUT, vol. 4, pp. 77:1–77:24 (2020)
18. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time-series motif from multi-dimensional data based on mdl principle. Mach. Learn. **58**, 269–300 (2005)
19. Fodor, I.K.: A survey of dimension reduction techniques. Technical report, LLNL, Livermore, USA (2002)

20. Zhu, Y., et al.: Exploiting a novel algorithm and GPUs to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins. KAIS **54**, 203–236 (2017)
21. Yeh, C.-C.M., Kavantzas, N., Keogh, E.: Matrix profile vi: meaningful multidimensional motif discovery. In: ICDM, pp. 565–574, IEEE (2017)
22. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR **7**, 1–30 (2006)

Comparing the Performance of Recurrent Neural Network and Some Well-Known Statistical Methods in the Case of Missing Multivariate Time Series Data

Samira Zahmatkesh(✉) and Philipp Zech

Department of Computer Science, University of Innsbruck, Innsbruck, Austria
{Samira.zahmat-kesh,Philipp.zech}@uibk.ac.at

Abstract. Missing values in multivariate time series data, often caused by network disruptions, device or power outages, and bad weather, can pose challenges for future analysis. Common statistical methods like multiple imputation and expectation-maximization are often used to impute missing time series data. However, these methods assume that the data is missing at random and may struggle with more complex missing data mechanisms and higher missing ratios. In these cases, advanced techniques like neural networks may offer improved imputation. This study assess the effectiveness of two recurrent neural network methods LSTM and GRU, enhanced with a time decay function, named LSTM-D and GRU-D, for analyzing missing multivariate time series. Their performance is compared with three well-known statistical methods: Bootstrapped-EM, EM-ARIMA, and MICE, across different missing data scenarios. Results indicate that LSTM-D and GRU-D perform better than traditional statistical methods for two different datasets, particularly when the missing data is not random.

Keywords: Time series · Missing data · Imputation · Neural networks · Statistical methods

1 Introduction

Time series data, characterized by its temporal nature with each data point linked to a specific timestamp, enables the analysis of changes over time. Multivariate Time Series (MVTS) includes two or more variables observed at the same time steps. This data is used in various domains, such as financial analysis, healthcare, manufacturing, environmental monitoring, and transportation.

The presence of missing data poses a challenge in the analysis and utilization of time series data due to factors such as sensor failures, network issues, human errors, non-response in surveys, system upgrades, or data storage limitations [34]. Understanding the reasons for missing values is crucial due to their significant impact on future analysis. According to Rubin [41] data with missing values can be categorized into three main groups, known as the missingness

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 35–51, 2025.
https://doi.org/10.1007/978-3-031-77066-1_3

mechanisms. Missing data mechanisms include Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR). In MCAR, missingness is unrelated to any data. In MAR, it depends on observed data. MNAR depends on unobserved data, making it the most challenging and complex mechanism, potentially leading to biases if not properly addressed. Handling missing data in time series analysis requires careful consideration of these mechanisms.

Common techniques for handling missing time series data include using complete case and imputation. Complete case or deletion is carried out either list-wise or pairwise [35], where mainly samples or variables are removed that are only partially observed. This method leaves gaps in the data set, possibly resulting in erroneous parameter estimations [22]. Imputation is predicting missing values based on available data. The choice of method depends on the nature of the missing data and aims. Many methods have been proposed based on statistical and machine learning models for imputing missing time series data.

Statistical imputation approaches have been mentioned as single imputation and Multiple Imputation (MI) methods. Single imputation is the process of filling distinctly one value for each missingness. Methods like mean/median averages [4], forward and backward imputation [45], or linear regression [53] are the most convenient single imputation methods that have been used, but there was significant bias and loss of precision. Also, the Auto-Regressive Moving Average Models (ARMA) [9] which is a combination of the basic linear processes, auto-regressive and moving average model, perform well in forecasting missing time series data. To address non-stationarity in time series data, Auto-Regressive Integrated Moving Average Models (ARIMA) are effective for imputing missing values. It is the integrated form of the ARMA that captures temporal patterns and trends for accurate data reconstruction [2,6,29,38]. The ARIMA model is used when the dataset exhibit temporal pattern and is of substantial volumes. However, alteration in observation and model specification leads the model to be unstable. Multiple imputation proposed by Rubin [33] which replaces missing values by $m \geq 2$ possible values, each with a unique estimate reflecting the uncertainty attached. The m estimates are combined to yield a single estimate. A wide variety of approaches based on MI have been proposed such as Expectation-Maximization (EM) [20], Probabilistic Matrix Factorization (PMF) [36], MI by Chained Equation (MICE) [49,54], Bayesian computational algorithm known as Markov Chain Monte Carlo (MCMC) [49]. Multiple imputation requires certain conditions to be met: data should be MAR, an appropriate imputation model should be used, and an adequate number of imputations should be performed to ensure convergence and address uncertainty.

ML methods like Bayesian network [47], support vector regression [57], and k-nearest neighbors [1], decision tree [21] have also been applied to the MVTS imputation problem. These methods are limited in covering complex temporal dependencies between observations [43]. Recently, various deep learning (DL) based methods have been introduced that are not only computationally feasible but also capable of addressing the complex missing data patterns in MVTS. The

Recurrent Neural Networks (RNN) like Gated Recurrent Unit (GRU) [14,16] and Long Short Term Memory (LSTM) [26,31] have the ability to represent temporal dependencies in sequential data and have been widely considered in analyzing MVTS. Moreover, they can capture long-range dependencies in time series data in an effective manner. Che et al., [14] proposed GRU-Decay (GRU-D) to handle missing values in medical data for classification purposes. A bidirectional RNN structure, which considers the input sequence in both forward and backward directions, based on LSTM [23] was presented by Cao et al. instead of the GRU-D to improve training accuracy [12]. Besides bi-directional RNN, Yoon et al., introduced the multi-directional RNN [58], which performs imputation across data streams. These models primarily address missingness in classification rather than prediction tasks. In multivariate time series, each variable may have different missingness ratios. Continuous and prolonged gaps, caused by sensor or equipment failures, pose challenges, leading to substantial information loss in consecutively correlated data. Therefore, a method capable of accurately reconstructing and analyzing such data is necessary. This study endeavors to analyze carefully the efficacy of two RNN-based models for imputation and prediction purposes. Specifically, we explore the performance of the GRU-D model proposed by Che et al. [14], and its extension to LSTM as LSTM-D. To implement these RNNs, we have used Python[1]. We have extended the code for the LSTM-D. Our objective is to compare their reconstruction capabilities for MVTS against three established statistical methods: the Bootstrapped-EM algorithm, the EM-ARIMA model, and multiple imputation using MICE, where we used reference implementations in R to conduct our experiments [50,56,59]. To better investigate our goal, we artificially create MAR, MNAR, and Long Gaps in the data. In the two latter cases, the missing data rate is deliberately high. In this paper, our primary objectives are outlined as follows:

- Assessing the performance the GRU-D and its extension to LSTM, known as LSTM-D in reconstructing MVTS under various rates and mechanisms of missingness.
- Assessing of three widely recognized statistical methods: Bootstrapped-EM, EM-ARIMA model, and MICE in imputation missing MVTS under various missing data scenarios.
- Conducting a comprehensive comparison between the performance of the RNNs and the selected statistical methods.

The rest of the paper is organized as follows. Section 2 gives an overview of RNN methods and details about LSTM-D as an extension of GRU-D. In Sect. 3 we present a brief introduction of the considered statistical methods. In Sect. 4 We implement the methods on air quality data and wind-turbin data. Finally, the conclusion and future works will be discussed in Sect. 5.

[1] The contributed source code link is "https://github.com/samirazahmat-kesh/GRU-D.git".

2 Recurrent Neural Networks for Multivariate Time Series Imputation

Recurrent neural networks, particularly LSTM [23] and GRU [16,17] architectures, are instrumental in multivariate time series prediction. LSTM and GRU are ideal for capturing dependencies in time-ordered data due to their memory cells and gating units, which address the vanishing gradient problem in traditional RNNs. These mechanisms enable selective retention and forgetting of information, crucial for long-term dependencies in time series analysis. They handle variable-length sequences and multivariate inputs effectively, making them versatile for various time series data. The choice between LSTM and GRU depends on model complexity and dataset characteristics. In essence, both excel in modeling sequential dependencies and predicting multivariate time series data.

When using RNNs for prediction and dealing with missing values, a straightforward approach involves replacing missing observations with the variable mean or assuming missing values are the same as their last measurement (forward imputation). However, these methods can result in loss of variability in the time series, assuming missing values have the same variability as observed ones, which may not be true, particularly with abrupt changes or fluctuations. Moreover, sensitivity to outliers, assuming constant time intervals, and artificially increasing variable correlation can lead to biased estimates. GRU-D, a variant of the GRU with a decay mechanism, was introduced by Che et al. [14] for clinical applications with a primary focus on classification tasks. The innovative GRU-D model improves sequential data analysis in medical contexts by integrating decay mechanisms into the standard GRU architecture. It demonstrates proficiency in handling missing data in clinical applications. This integration enhances sequence modeling and the model's robustness in managing incomplete or missing information in real-world medical datasets. An extension of GRU-D to LSTM, termed LSTM-D, is presented in this paper.

LSTM unit is made of the memory cell, which stores information over periods. The cell is controlled by three gates: the input gate ($\mathbf{i}$), the forget gate ($\mathbf{f}$), and the output gate ($\mathbf{o}$). These gates are responsible for regulating the flow of information into, out of, and within the memory cell. The input gate determines how much of the new information should be stored in the cell, the forget gate controls the extent to which the existing information is retained, and the output gate manages the information to be outputted to the next time step. By carefully adjusting these gates, LSTM can capture and preserve relevant information over long sequences, making it a powerful tool for tasks such as natural language processing, speech recognition, and time-series analysis. An LSTM unit also has a hidden state represented by $\mathbf{h}_{t-1}$ and $\mathbf{h}_t$ for the hidden state of the previous time stamp and the current timestamp, respectively. And has a cell state represented by $\mathbf{c}_{t-1}$ and $\mathbf{c}_t$ for the previous and current timestamps, respectively. Hidden state is known as short term memory, and the cell state is known as Long term memory.

We denote a multivariate time series with D variables observed at T times $X = (\mathbf{x}_1, ..., \mathbf{x}_T)^T \in \mathbb{R}^{T \times D}$, thus x_t^d is the sample for d-th variable at time t, and

$\mathbf{x}_t = (x_t^1, ..., x_t^D)$ is the measurement vector at time t for all the variables, where $t = 1, \ldots, T$ and $d = 1, \ldots, D$. The equations for LSTM gates are

$$\begin{aligned}
\mathbf{f}_t &= \sigma(W_f \mathbf{h}_{t-1} + R_f \mathbf{x}_t + b_f) \\
\mathbf{i}_t &= \sigma(W_i \mathbf{h}_{t-1} + R_i \mathbf{x}_t + b_i) \\
\mathbf{o}_t &= \sigma(W_o \mathbf{h}_{t-1} + R_o \mathbf{x}_t + b_o) \\
\tilde{\mathbf{c}}_t &= \phi(W_c \mathbf{h}_{t-1} + R_c \mathbf{x}_t + b) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \phi(\mathbf{c}_t)
\end{aligned}$$

where σ and ϕ stand for activation functions which often are considered sigmoid and tanh, respectively. W_f, W_f, W_f, R_f, R_i and R_o are the relevant weight matrices, b_f, b_i, b_o and b are the bias vectors, and all of them are as the model parameters. The cell architecture for LSTM is shown in Fig. 1(a). To effectively address missing values in MVTS, LSTM-D employs a dynamic decay mechanism, adept at capturing temporal dependencies and mitigating the impact of incomplete data. To denote which measurement is missing or observed, we define a masking matrix $\mathbb{M}$ with the same dimension of X with the elements m_t^d as:

$$m_t^d = \begin{cases} 1 & x_t^d \text{ is observed} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

To track missing values for each variable in X, the last time interval is maintained in a matrix $\Delta \in R^{T \times D}$ with elements $\delta_t^d \in R$ as

$$\delta_t^d = \begin{cases} s_t - s_{t-1} + \delta_{t-1}^d, & \text{if } t > 1, m_{t-1}^d = 0 \\ s_t - s_{t-1}, & \text{if } t > 1, m_{t-1}^d = 1 \\ 0, & \text{if } t = 1 \end{cases} \tag{2}$$

where s_t are the time stamps relative to each measurement. A vector of decay rates, γ is defined as

$$\gamma_t = \exp\{-\max(0, W_\gamma \delta_t + b_\gamma)\}, \tag{3}$$

W_γ and b_γ are also the model parameters. This decay rate will be applied to the input as γ_x and to the hidden state as γ_h. Thus the input valus are updated as

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\gamma_{xt}^d x_{t'}^d + (1 - m_t^d)(1 - \gamma_{xt}^d)\widetilde{x}^d \tag{4}$$

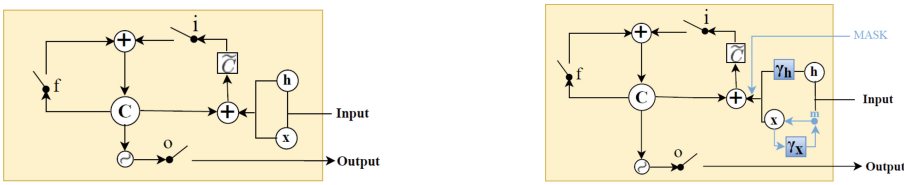
where $x_{t'}^d$ is the last observed value and the $\widetilde{x}^d$ is the empirical mean of d-th variable. Also, to capture better the information of missingness, the previous hidden state $\mathbf{h}_{t-1}$ is decayed before computing the new hidden state h_t as

$$\mathbf{h}_{t-1} \leftarrow \mathbf{h}_{t-1} \odot \gamma_{ht}. \tag{5}$$

Moreover the masking vectors ($\mathbf{m}_t$) are fed directly into the model. Thus, the modification of LSTM with decay mechanism as LSTM-D will be with the following equations over the cell memory and all the gates:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(W_f \mathbf{h}_{t-1} + R_f \mathbf{x}_t + V_f \mathbf{m}_t + b_f) \\
\mathbf{i}_t &= \sigma(W_i \mathbf{h}_{t-1} + R_i \mathbf{x}_t + V_i \mathbf{m}_t + b_i) \\
\mathbf{o}_t &= \sigma(W_o \mathbf{h}_{t-1} + R_o \mathbf{x}_t + V_o \mathbf{m}_t + b_o) \\
\tilde{\mathbf{c}}_t &= \phi(W_c \mathbf{h}_{t-1} + R_c \mathbf{x}_t + V \mathbf{m}_t + b) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \phi(\mathbf{c}_t)
\end{aligned}$$

where V_f, V_i and V_o are the new added parameters. Figure 1(b).



(a) (b)

Fig. 1. The architecture of LSTM and LSTM-D cell

3 Well-Known Statistical Methods for Multivariate Time Series Imputation

Various methods for multivariate time series imputation have been introduced, including regression-based, auto-regressive, and Bayesian approaches. These methods provide robust solutions for handling missing data in complex temporal datasets by leveraging statistical relationships and dependencies among variables. In our study, we focus on three effective methods: Bootstrap-based EM, EM-ARIMA, and MICE, which we briefly introduce here.

3.1 Bootstrap-Based EM

In general, the Bootstrap-based EM algorithm draws m (the number of imputation datasets) samples of size n (the size of the original dataset) from original datasets, and point estimates of mean and variance are performed in each sample using the EM method. Let $X = (\mathbf{x}_1, ..., \mathbf{x}_D)^T \in \mathbb{R}^{n \times D}$ be the data matrix with observed part X^{obs} and unobserved part X^{mis} and has a multivariate normal distribution as $X \sim \mathrm{N}_D(\mu, \Sigma)$. Also, It is assumed data are MAR. Similar

to Sect. 2, Let M be the missingness mask matrix. Under MAR assumption $p(M|X) = P(M|X^{obs})$, then the joint distribution of Observed data and mask can be broken up as

$$p(X^{obs}, M|\theta) = p(M|X^{obs})p(X^{obs}|\theta)$$

As the inferences are based on the complete data parameters, the likelihood can be written as

$$L(\theta|X^{obs}) \propto p(X^{obs}|\theta)$$

which can be rewritten using the law of iterated expectations [42] as

$$L(\theta|X^{obs}) \propto p(X^{obs}|\theta) = \int p(X|\theta)dX^{mis}.$$

The main computational difficulty in the analysis of incomplete data is taking draws from this posterior. The EM algorithm [20] is a simple computational approach to finding the mode of the posterior. Bootstrap-based EM algorithm combines the classic EM algorithm with a bootstrap approach to take draws from this posterior. For each draw, the data are bootstrapped to simulate estimation uncertainty and then the EM algorithm is run to find the mode of the posterior for the bootstrapped data, which gives fundamental uncertainty too [24]. After drawing the posterior of the complete-data parameters, imputations take place by drawing values of X^{mis} from its distribution conditional on X^{obs} and the draws of θ. Remember there are m sets of estimates. Then each set of estimates is used to impute the missing observations from original dataset. The result is m sets of imputed data that can be used for subsequent analyses.

3.2 EM-ARIMA

As the previous section, let $\mathbf{x}_t$ be the tth realization of a D-variates time series. If l components of $\mathbf{x}_t$ are unobserved, then it can be rearranged and divided in two missing and observed parts, which are denoted by $\mathbf{x}_t = (\mathbf{x}_{tm}, \mathbf{x}_{to})$. Also It is assumed X is of multivariate normal distribution with mean μ and covariance matrix Σ. Then they also can be partitioned as following:

$$\tilde{\mu}_t = \left[\tilde{\mu}_{tm}\ \tilde{\mu}_{to}\right] \qquad \text{and} \qquad \tilde{\Sigma}_t = \begin{bmatrix} \tilde{\Sigma}_{t(mm)} & \tilde{\Sigma}_{t(mo)} \\ \tilde{\Sigma}_{t(om)} & \tilde{\Sigma}_{t(oo)} \end{bmatrix}.$$

The method is proposed by Junger and Leon [25], and is a modification of EM algorithm [20]. In the imputation algoritm, first the missing values are replaced by some initial estimates and the parameters μ and Σ are estimated. Then, the level for each of the uni-variate time series is estimated by ARIMA model, and finally re-estimate the missing values using updated estimates of the parameters and the level of the time series. These steps are iterated until some convergence criterion is reached. At the $(k + 1)$-th iteration of the E (estimation) step of

the EM algorithm, the missing values are imputed with conditional means given the observed values and the previous estimates of the parameters given by

$$\begin{aligned}\tilde{\mathbf{x}}_{tm}^{(k+1)} &= \mathrm{E}\left[\mathbf{x}_{tm} |\ \mathbf{x}_{to}, \tilde{\mu}_t^{(k)}, \tilde{\Sigma}_t^{(k)}\right] \\ &= \tilde{\mu}_{tm}^{(k)} + \tilde{\Sigma}_{t(mo)}^{(k)} \tilde{\Sigma}_{t(oo)}^{(k)} \left(\mathbf{x}_{to} + \tilde{\mu}_{to}^{(k)}\right).\end{aligned}$$

In the M (maximization) step, the revised maximum likelihood estimates of μ_t and Σ_t are computed. As we mentioned, The temporal contribution to the level of each time series μ_t is independently estimated using ARIMA model [7].

3.3 MICE

A popular approach for implementing multiple imputation is sequential regression modeling, also called multiple imputation by chained equations (MICE) [11,39]. The basic idea in MICE is imputing missing values in one variable from a regression of the observed elements of it condition on the other variables. Let $X = (\mathbf{x}_1, ..., \mathbf{x}_D)$ be the data matrix consists of D variables. Each variable may be partially observed so we divide the vector of each variable into two parts: observed (obs) and missing (mis) as $\mathbf{x}_d = (\mathbf{x}_d^{obs}, \mathbf{x}_d^{mis})$. The imputation problem is to draw the unconditional multivariate distribution of X from $P(X)$. Let n denote an iteration counter, one may repeat the following sequence of Gibbs sampler iterations:

$$\begin{aligned}&\text{For } \mathbf{x}_1: \quad \text{draw } \mathbf{x}_1^{n+1} \text{ from } P(\mathbf{x}_1 | \mathbf{x}_2^n, \mathbf{x}_3^n, ..., \mathbf{x}_D^n) \\ &\text{For } \mathbf{x}_2: \quad \text{draw } \mathbf{x}_2^{n+1} \text{ from } P(\mathbf{x}_2 | \mathbf{x}_1^{n+1}, \mathbf{x}_3^n, ..., \mathbf{x}_D^n) \\ &\vdots \\ &\text{For } \mathbf{x}_D: \quad \text{draw } \mathbf{x}_D^{n+1} \text{ from } P(\mathbf{x}_D | \mathbf{x}_1^{n+1}, \mathbf{x}_2^{n+1}, ..., \mathbf{x}_{D-1}^{n+1})\end{aligned}$$

where means condition each time on the most recently drawn values of all other variables the candidate variable is imputed. This can be performed for l time until convergence. All the procedure will be repeated m times, yielding m imputed sets which then the best imputed set can be selected based on an appropriate criterion. Using $l = 10$ typically yields satisfactory results. It is standard to use generalized linear models as the basis of the predictive draws, but other kind of models also have been applied. the details about specification of the imputation model or monitoring convergence can be found in references [10,11].

4 Results and Discussion

In this section, we first introduce the data used in the experiment and explain how to implement different scenarios of missingness in them. Then the obtained results in imputation missing data with two RNNs and three statistical methods are presented. Finally, we have a comprehensive discussion about the results and the future works.

4.1 Experiments

In this study, we employed two distinct datasets: air quality data and wind-turbin data. Air quality data [52] is used as the basis for our analyses, while the wind-turbin data (from kaggle.com), was utilized to independently confirm and ensure the robustness of our findings, enhancing the reliability and generalizability of our research.

The air quality data set contains 8784 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multi-sensor Device. The device was located on the field in a significantly polluted area, at road level, within an Italian city. Data were recorded from March 2004 to March 2005 (one year) representing the longest freely available recordings of on field deployed air quality chemical sensor responses. The dataset consists of the hourly averaged of ten air quality variables. In this study, we have chosen five of them for our prediction and imputation purpose, which are: PT08.S1 (tin oxide, which is nominally CO targeted) PT08.S2 (titania, which is nominally NMHC[2] targeted), PT08.S3 (tungsten oxide, which is nominally NOx targeted), PT08.S4 (tungsten oxide, which is nominally NO2 targeted) and PT08.S5 (indium oxide, which is nominally O3 targeted). As the amount of missing values are limited, we have imposed artificially two missing data mechanisms MAR and MNAR, and also, we have created long gaps during the data so that every variable is missing for consecutive days of a month or more. In order to create MAR in each variable with specific missing rate p, we generated a random binary matrix M as Eq. (1) with probability of being zero equal to p and probability of being one equal to 1-p for each m_t^d. Then we replace x_t^d with NA wherever $m_t^d = 0$. To creat MNAR in the data with different missing rates, we considered the median of each variables in X and if x_t^d be the observation of dth variable at time t then:

$$x_t^d = \begin{cases} Missing & \text{if } x_t^d > a + c * median \\ Observed & \text{otherwise} \end{cases} \tag{6}$$

We considered (a,c) equal to (−0.1,1), (0,0.9), (−0.1,1), (0,1) and (−0.1,1) for variables CO, NMHC, NO_x, NO_2 and O_3 respectively. Furthermore, to insert long gaps in the data, we chose randomly some periods for each variable and replaced the data with NAs. In the event of Long Gaps, some points are also MAR inherently. In Table 1 the rates of missing data in each variable under the specific type of missingness are reported.

The second dataset, wind-turbin data, consists of measurements of 19 variables from Jan 2018 to Mar 2020 of a wind turbine, at a 10 min frequency. Here, we use data from Jan 2018 to Dec 2018, with a total 52704 observation. We use four variables "Active Power", "Ambient temperature", "Wind direction" and "Wind Speed" as input features. There are missing values in all variables, but in order to fulfill the assumption of MNAR with higher missing rates, we manipulated the data similar to air quality data and created MNAR with missing

[2] Non-methane Hydrocarbons.

rates 57, 58,79 and 67% respectively in “Active Power”, “Ambient temperature”, “Wind direction” and “Wind Speed” variables.

Table 1. Missing rates of air-quality dataset variables for different kind of missingness.

Variable	Missing Rates (%)	Missing type
CO	40	MAR
NMHC	57.7	
NO_x	36.5	
NO_2	14	
O_3	28	
CO	70.5	MNAR
NMHC	63.2	
NO_x	64	
NO_2	53.2	
O_3	60.2	
CO	49.6	Long Gaps
NMHC	64.5	
NO_x	42.5	
NO_2	34	
O_3	41.3	

It is worth to note that we normalize the time series data in each dataset before using it in each method. We do this by using max-min normalization, which scales all the variables to a range between 0 and 1. The parameters for LSTM-D and GRU-D model are set as: learning rate = 0.01, batch size = 64, optimizer = Adam, Epoch = 1000 and time step = 10. The EM-bootstrapping algorithm has been implemented via a multiple imputation with m = 2000 iterations and ultimately the imputation with least error has been selected. We have considered EM-ARIMA model with several Parameter values for p, d and q, eventually ARIMA(1,1,1) was selected and reported here. The performance of each method is reported in aspect of the averaged mean absolute error (MAE) and averaged mean square error (MSE) in Table 2, where each one is calculated as follow:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

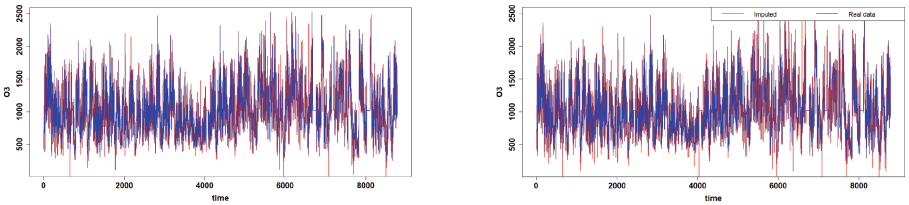
4.2 Results

In MAR case the accuracies of all the methods are very close to each other and near zero, so that they cannot be distinguished and all of them are doing well. In MNAR case, the results show the better performance of LSTM-D and GRU-D, with a significant difference in the MAE and MSE values compared to statistical methods. In Long Gaps Case, the two RNN methods performs again better. Of course, it cannot be concluded that this superiority is very high compared to statistical methods. In this case, among statistical methods, EM-ARIMA model has better performance with lower MSE and MAE, and is doing as well as LSTM-D and GRU-D. Therefore, according to what we checked, considered RNNs are more capable of handling non-random missingness and long gaps than the well-known statistical methods, although this issue requires further investigation in other applications and in the case of other patterns of missingness. Since, the results for all the variables are equivalent, we have chosen "O3" and plot the real data and imputed data together in Fig. 2 and 3. Obviously, the imputed data is almost identical with real data for LSTM-D and GRU-D methods, but for the three statistical methods especially with EM-Bootstrap and MICE, in some periods compliance has not been achieved.

Table 2. Result of different methods in reconstruction air-quality data-set variables, under three type of missingness assumption

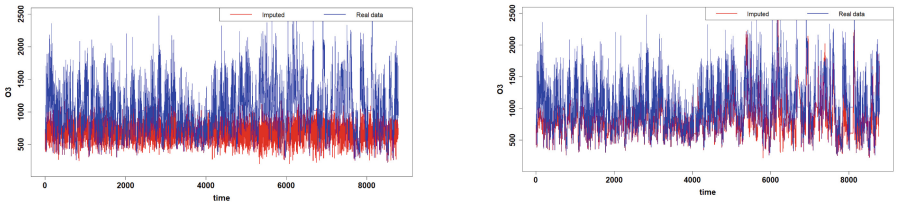
Method	Train MSE	Test MSE	Test MAE	Missing Type
GRU-D	0.029	0.030	0.139	MAR
LSTM-D	0.048	0.048	0.187	
EM-Bootstrap	-	0.161	0.184	
EM-ARIMA	-	0.030	0.074	
MICE	-	0.163	0.186	
GRU-D	0.071	0.067	0.176	MNAR
LSTM-D	0.058	0.060	0.152	
EM-Bootstrap	-	1.605	1.033	
EM-ARIMA	-	1.642	1.005	
MICE	-	1.690	1.047	
GRU-D	0.033	0.033	0.137	Long Gaps
LSTM-D	0.024	0.025	0.118	
EM-Bootstrap	-	0.256	0.267	
EM-ARIMA	-	0.081	0.184	
MICE	-	0.279	0.282	

Furthermore, the MAE and MSE of different methods for reconstruction Wind-Turbin data variables are reported in Table 3. The two RNNs performs almost similarly, and among statistical methods EM-ARIMA performs better.

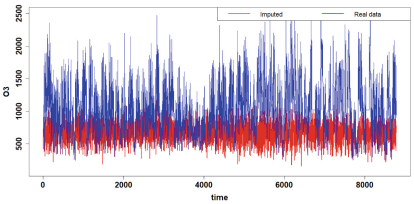


(a) LSTM-D (b) GRU-D

Fig. 2. Reconstruction (prediction) of the variable O3 in MNAR case with LSTM-D and GRU-D



(a) EM-Bootstrapping (b) EM-ARIMA



(c) MICE

Fig. 3. Reconstruction (prediction) of the variable O3 in MNAR case with statistical methods: EM-ARIMA, EM-Bootstrap and MICE methods

It can be seen that the error of the two RNNs is much less than the statistical methods.

4.3 Discussion

In this paper we compared the performance of two modified RNNs (LSTM-D and GRU-D) and three statistical methods for imputing missing multivariate time series (MVTS) data. The study focused on different missingness scenarios: Missing At Random (MAR), Missing Not At Random (MNAR), and Long Gaps, which were artificially applied to the data. In tests with air quality data, the RNNs generally performed better than statistical methods, particularly as the missing ratio increased, with MNAR showing significant improvement. Then, to ensure the stability of the result, we experimented on the wind turbine data and the results confirmed the superior performance of the RNNs as well.

Table 3. Result of different methods for reconstruction Wind-Turbin data variables

Method	Train MSE	Test MSE	Test MAE
GRU-D	0.028	0.029	0.077
LSTM-D	0.029	0.030	0.080
EM-Bootstrap	-	2.174	1.179
EM-ARIMA	-	1.403	0.939
MICE	-	2.334	1.229

Many time series data exhibit both temporal and spatial dependencies, known as spatio-temporal data. Traditional methods like KNN-based methods [15,48], EM [20], singular value decomposition [46], random forest [8] often treat spatial and temporal dimensions separately. Oversimplified methods can lead to sub-optimal results, especially with complex, dynamic scenarios and significant missing values in spatio-temporal data. Various statistical and ML/DL approaches have been proposed to address these challenges by leveraging spatial and temporal correlations to improve missing data for classification or prediction [5,18,19,28,30,32,44,51].

DL methods can generate accurate predictions by leveraging complex missing data mechanisms and dependencies but have flaws [3]. Despite their power, DL methods have limitations in statistical modeling for spatial and spatio-temporal data due to substantial uncertainties caused by inherent variability such as measurement errors or natural fluctuations, data gaps, mismatched prediction supports, and sampling. They also do not directly estimate prediction or classification errors and struggle to incorporate known mechanistic relationships in spatio-temporal data.

That is problematic when one is attempting to use the output from these models to make decisions, it is challenging as it is not obvious how much reliable the output is. Also, a more troubling issue is that most DL methods are "black boxes" and it is difficult to know why they are producing the prediction or classification that they do, meaning that their internal workings are not easily interpretable or explainable by humans [40].

In opposite, in traditional statistical models, such as linear regression, it is relatively easy to interpret the impact of each variable on the model's output. One can understand how changes in input variables relate to changes in the output. However, the main issue with these methods is that there may be cases where they are not sufficient to capture the broad nature of spatial non-linearity and complex features inside data. In some cases, data may exhibit complex multivariate features, non-Gaussianity, non-stationarity, and MNAR or any complex missingness patterns. Considering these cases, the spatio-temporal estimation of continuous variables could be a challenging task. DL models allow one to identify the patterns in the complex datasets and to make estimations/predictions based on them. The key feature of the DL models is that they learn from the data,

and they do not require rigid statistical assumptions (such as stationarity and linearity) [55].

In order to mitigate the shortcomings of each of the DL-based and statistical-based methods, there has been an increasing number of works in recent years, that take a hybrid approach for analyzing different kinds of data where spatio-temporal data were not exempted, for example: [13,27,37]. These hybrid models borrow some of the effective ideas from each DL methods and statistical models in order to facilitate modeling the data, produce uncertainties for model outputs and enhance the interpretability of DL methods. Thus, they strike a balance between predictive accuracy and the ability to understand and trust the model's decisions. Despite promising efforts, our investigation reveals a significant gap in the availability of a hybrid framework that ensures high prediction/imputation accuracy, model-based uncertainty quantification, and explainability for missing spatio-temporal data across various applications. As a hypothesis, the fusion of DL methods with classical statistical models may offer an attractive way forward. Since this issue has not been addressed in the literature, working on developing a novel hybrid method in the case of complex missingness patterns and mechanisms in spatio-temporal data is in our future work plan. We intend to present a comprehensive and hybrid model that in one hand uses the advantages of statistical methods to model the missing process and the measurement process together so that it is possible to explain the relationships between variables and missingness processes altogether in a spatio-temporal model framework with the ability of uncertainty quantification, and on the other hand, by using a suitable neural network, the complexities in the spatio-temporal data can be modeled with the aim of prediction/imputation with higher accuracy.

5 Conclusion

This study demonstrates that GRU-D and LSTM-D, modified RNN methods with a decay mechanism, outperform traditional statistical methods (Bootstrapped-EM, EM-ARIMA, and MICE) in reconstructing missing multivariate time series data, especially under high rates of missingness with MNAR or long gaps. Statistical methods have been used over the years and can produce interpretable results and quantify uncertainties, while deep learning methods do not have this ability due to being "black boxes" [3]. Our study paves the way for our future research, aiming to develop a hybrid framework combining deep learning and statistical methods. This approach will seek to enhance the neural networks-based method's effectiveness in handling missing data with complex correlations, including spatio-temporal data.

References

1. Acuna, E., Rodriguez, C.: The treatment of missing values and its effect on classifier accuracy. In: Classification, Clustering, and Data Mining Applications: Proceedings of the Meeting of the International Federation of Classification Societies (IFCS),

Illinois Institute of Technology, Chicago, 15–18 July 2004, pp. 639–647. Springer (2004)
2. Afrifa-Yamoah, E., Mueller, U.A., Taylor, S., Fisher, A.: Missing data imputation of high-resolution temporal climate time series data. Meteorol. Appl. **27**(1), e1873 (2020)
3. Alzubaidi, L., et al.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J. Big Data **8**, 1–74 (2021)
4. Aydilek, I.B., Arslan, A.: A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. Inf. Sci. **233**, 25–35 (2013)
5. Beckers, J.M., Rixen, M.: EOF calculations and data filling from incomplete oceanographic datasets. J. Atmos. Oceanic Tech. **20**(12), 1839–1856 (2003)
6. Box, G.E., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control, vol. 734. Wiley (2011)
7. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. Wiley (2015)
8. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
9. Broersen, P.M., Bos, R.: Time-series analysis if data are randomly missing. IEEE Trans. Instrum. Meas. **55**(1), 79–84 (2006)
10. Burgette, L.F., Reiter, J.P.: Multiple imputation for missing data via sequential regression trees. Am. J. Epidemiol. **172**(9), 1070–1076 (2010)
11. Buuren, S.V., Oudshoorn, K.: Flexible multivariate imputation by mice (1999)
12. Cao, W., Wang, D., Li, J., Zhou, H., Li, L., Li, Y.: Brits: bidirectional recurrent imputation for time series. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
13. Chattopadhyay, A., Mustafa, M., Hassanzadeh, P., Bach, E., Kashinath, K.: Towards physics-inspired data-driven weather forecasting: integrating data assimilation with a deep spatial-transformer-based u-net in a case study with era5. Geoscientific Model Dev. **15**(5), 2221–2237 (2022)
14. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Sci. Rep. **8**(1), 6085 (2018)
15. Chen, J., Shao, J.: Nearest neighbor imputation for survey data. J. Official Stat. **16**(2), 113 (2000)
16. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
17. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
18. Ciampi, A., Appice, A., Guccione, P., Malerba, D.: Integrating trend clusters for spatio-temporal interpolation of missing sensor data. In: Web and Wireless Geographical Information Systems: 11th International Symposium, W2GIS 2012, Naples, Italy, 12–13 April 2012. Proceedings 11, pp. 203–220. Springer (2012)
19. Cui, Z., Lin, L., Pu, Z., Wang, Y.: Graph Markov network for traffic forecasting with missing data. Transp. Res. Part C: Emerg. Technol. **117**, 102671 (2020)
20. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc.: Ser. B (Methodol.) **39**(1), 1–22 (1977)
21. Fouladgar, N., Främling, K.: A novel LSTM for multivariate time series with massive missingness. Sensors **20**(10), 2832 (2020)
22. Graham, J.W.: Missing data analysis: making it work in the real world. Annu. Rev. Psychol. **60**, 549–576 (2009)

23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
24. Honaker, J., King, G.: What to do about missing values in time-series cross-section data. Am. J. Polit. Sci. **54**(2), 561–581 (2010)
25. Junger, W., De Leon, A.P.: Imputation of missing data in time series for air pollutants. Atmos. Environ. **102**, 96–104 (2015)
26. Kim, Y.J., Chi, M.: Temporal belief memory: imputing missing data during RNN training. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018) (2018)
27. Kirkwood, C., Economou, T., Pugeault, N.: Bayesian deep learning for mapping via auxiliary information: a new era for geostatistics? arXiv preprint arXiv:2008.07320 (2020)
28. Kondrashov, D., Ghil, M.: Spatio-temporal filling of missing points in geophysical data sets. Nonlinear Process. Geophys. **13**(2), 151–159 (2006)
29. Layanun, V., Suksamosorn, S., Songsiri, J.: Missing-data imputation for solar irradiance forecasting in Thailand. In: 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp. 1234–1239. IEEE (2017)
30. Le, T.T., Le Nguyen, P., Binh, H.T.T., Akerkar, R., Ji, Y., et al.: Gcrint: network traffic imputation using graph convolutional recurrent neural network. In: ICC 2021-IEEE International Conference on Communications, pp. 1–6. IEEE (2021)
31. Lee, M., An, J., Lee, Y.: Missing-value imputation of continuous missing based on deep imputation network using correlations among multiple IoT data streams in a smart space. IEICE Trans. Inf. Syst. **102**(2), 289–298 (2019)
32. Li, Y., Parker, L.E.: Nearest neighbor imputation using spatial-temporal correlations in wireless sensor networks. Inf. Fusion **15**, 64–79 (2014)
33. Little, R., Rubin, D.: Multiple Imputation for Nonresponse in Surveys. Wiley **10**, 9780470316696 (1987)
34. Little, R.J., Rubin, D.B.: Statistical Analysis with Missing Data, vol. 793. Wiley (2019)
35. McKnight, P.E., McKnight, K.M., Sidani, S., Figueredo, A.J.: Missing Data: A Gentle Introduction. Guilford Press (2007)
36. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, vol. 20 (2007)
37. Mohan, A.T., Lubbers, N., Livescu, D., Chertkov, M.: Embedding hard physical constraints in convolutional neural networks for 3D turbulence. In: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, vol. 520 (2020)
38. Moritz, S., Sardá, A., Bartz-Beielstein, T., Zaefferer, M., Stork, J.: Comparison of different methods for univariate time series imputation in R. arXiv preprint arXiv:1510.03924 (2015)
39. Raghunathan, T.E., Lepkowski, J.M., Van Hoewyk, J., Solenberger, P., et al.: A multivariate technique for multiply imputing missing values using a sequence of regression models. Surv. Methodol. **27**(1), 85–96 (2001)
40. Reichstein, M., et al.: Deep learning and process understanding for data-driven earth system science. Nature **566**(7743), 195–204 (2019)
41. Rubin, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (1976)
42. Schervish, M.J., DeGroot, M.H.: Probability and Statistics, vol. 563. Pearson Education, London (2014)
43. Siami-Namini, S., Tavakoli, N., Namin, A.S.: A comparative analysis of forecasting financial time series using Arima, LSTM, and BiLSTM. arXiv preprint arXiv:1911.09512 (2019)

44. Song, C., Yang, X., Shi, X., Bo, Y., Wang, J.: Estimating missing values in China's official socioeconomic statistics using progressive spatiotemporal Bayesian hierarchical modeling. Sci. Rep. **8**(1), 10055 (2018)
45. Song, S., Li, C., Zhang, X.: Turn waste into wealth: On simultaneous clustering and cleaning over dirty data. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1115–1124 (2015)
46. Stewart, G.W.: On the early history of the singular value decomposition. SIAM Rev. **35**(4), 551–566 (1993)
47. Susanti, S.P., Azizah, F.N.: Imputation of missing value using dynamic Bayesian network for multivariate time series data. In: 2017 International Conference on Data and Software Engineering (ICoDSE), pp. 1–5. IEEE (2017)
48. Tak, S., Woo, S., Yeo, H.: Data-driven imputation method for traffic data in sectional units of road links. IEEE Trans. Intell. Transp. Syst. **17**(6), 1762–1771 (2016)
49. Takahashi, M.: Statistical inference in missing data by MCMC and non-MCMC multiple imputation algorithms: assessing the effects of between-imputation iterations. Data Sci. J. **16**, 37 (2017)
50. Van Buuren, S., Groothuis-Oudshoorn, K.: mice: multivariate imputation by chained equations in R. J. Stat. Softw. **45**, 1–67 (2011)
51. Venturi, D., Karniadakis, G.E.: Gappy data and reconstruction procedures for flow past a cylinder. J. Fluid Mech. **519**, 315–336 (2004)
52. Vito, S.: Air Quality. UCI Machine Learning Repository (2016)
53. Von Hippel, P.T.: 4. regression with missing ys: an improved strategy for analyzing multiply imputed data. Sociol. Methodol. **37**(1), 83–117 (2007)
54. White, I.R., Royston, P., Wood, A.M.: Multiple imputation using chained equations: issues and guidance for practice. Stat. Med. **30**(4), 377–399 (2011)
55. Wikle, C.K., Zammit-Mangion, A.: Statistical deep learning for spatial and spatiotemporal data. arXiv preprint arXiv:2206.02218 (2022)
56. Wu, R., Hamshaw, S.D., Yang, L., Kincaid, D.W., Etheridge, R., Ghasemkhani, A.: Data imputation for multivariate time series sensor data with large gaps of missing data. IEEE Sens. J. **22**(11), 10671–10683 (2022)
57. Wu, S.F., Chang, C.Y., Lee, S.J.: Time series forecasting with missing values. In: 2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom), pp. 151–156. IEEE (2015)
58. Yoon, J., Zame, W.R., van der Schaar, M.: Estimating missing data in temporal data streams using multi-directional recurrent neural networks. IEEE Trans. Biomed. Eng. **66**(5), 1477–1490 (2018)
59. Zhang, Z.: Multiple imputation for time series data with Amelia package. Ann. Transl. Med. **4**(3) (2016)

Accurate and Efficient Real-World Fall Detection Using Time Series Techniques

Timilehin B. Aderinola[1(✉)], Luca Palmerini[2], Ilaria D'Ascanio[2], Lorenzo Chiari[2], Jochen Klenk[3], Clemens Becker[3,4], Brian Caulfield[1], and Georgiana Ifrim[1]

[1] Insight SFI Research Centre for Data Analytics, University College Dublin, Dublin, Ireland
{timilehin.aderinola,b.caulfield,georgiana.ifrim}@ucd.ie
[2] Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, Bologna, Italy
{luca.palmerini,ilaria.dascanio2,lorenzo.chiari}@unibo.it
[3] Department of Clinical Gerontology, Robert Bosch Hospital, Stuttgart, Germany
{Jochen.Klenk,clemens.becker}@rbk.de
[4] Digital Geriatrics Unit, Heidelberg University Hospital, Heidelberg, Germany

Abstract. Falls pose a significant health risk, particularly for older people and those with specific medical conditions. Therefore, timely fall detection is crucial for preventing fall-related complications. Existing fall detection methods often have high false alarm or false negative rates, and many rely on handcrafted features. Additionally, most approaches are evaluated using simulated falls, leading to performance degradation in real-world scenarios. This paper explores a new fall detection approach leveraging real-world fall data and state-of-the-art time series techniques. The proposed method eliminates the need for manual feature engineering and has efficient runtime. Our approach achieves high accuracy, with false alarms and false negatives each as few as one in three days on FARSEEING, a large dataset of real-world falls (mean F_1 score: 90.7%). We also outperform existing methods on simulated falls datasets, FallAllD and SisFall. Furthermore, we investigate the performance of models trained on simulated data and tested on real-world data. This research presents a real-time fall detection framework with potential for real-world implementation.

Keywords: Fall detection · Time series · Real-world falls

1 Introduction

A fall is "an event which results in a person coming to rest inadvertently on the ground, floor, or other lower level". Globally, falls cause more than 684,000 deaths and about 172 million disabilities annually [52]. Most falls go unreported, especially those without injuries [40]. However, older adults [50] and people with specific medical conditions are considered high-risk groups (HRG) because

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 52–79, 2025.
https://doi.org/10.1007/978-3-031-77066-1_4

they fall more frequently [31] and may be unable to alert anyone if they sustain fall-related injuries. Hence, falls among HRGs can significantly affect their livelihood, instilling a fear of falling, threatening their independence, and posing life-threatening risks. The ageing global population increases the strain on healthcare systems from falls, due to the need for hospitalisation, medical care, and potential surgery [4]. Consequently, extensive research has been dedicated to automated fall detection for timely intervention to prevent fall-related complications, such as permanent disability or mortality.

Fall detection typically involves data capture, preprocessing, feature extraction, and classification [24]. The data capture stage records participants' daily activities to identify both normal activities of daily living (ADLs) and falls. Fall data can be collected using vision-based techniques, ambient devices, and wearable sensors. Recent vision-based techniques leverage human pose estimation for fall detection using body geometry features in videos [2]. However, vision-based techniques face challenges like privacy concerns and high storage requirements. Ambient devices, while suitable for private areas like bathrooms, are expensive and may have limited coverage [24]. Wearable devices offer a practical solution for capturing real-world falls since they are relatively low-cost and can be carried for extended periods [20] in both indoor and outdoor settings.

Fall detection algorithms analyse motion data to distinguish between falls and ADLs using threshold-based, machine learning based, or deep learning based approaches [24]. Threshold-based methods [21,47] use cut-off values set on the sensor signals to distinguish between falls and ADLs. However, these techniques commonly have high false alarm rates [42], which could lead to "false alarm fatigue" [32]. On the other hand, machine learning-based methods such as [22,46] use conventional classifiers with manually crafted features, yet, modelling the diverse factors that lead to falls is challenging. These factors can be internal (age, impaired mobility, disease), external (lighting, obstacles), or situational (activity leading to the fall). Consequently, fall feature extraction is complex and often requires a multidisciplinary approach [50].

Recently, deep learning approaches [6,23,25,29,36,53] have been used for fall detection, removing the need for manual feature extraction. While these methods show promising results, most are trained and tested on simulated data due to the scarcity of real fall data, raising concerns about their applicability in real-world settings [48]. Furthermore, deep learning models require large amounts of representative data, long training times, with a large number of parameters [15]. Hybrid methods [14,25,51] combining thresholds, machine learning, and deep learning have been proposed to improve accuracy, but often at the expense of simplicity and efficiency.

A major limitation of most fall detection approaches is their lack of transferability to real-world scenarios. Studies [38,45] have shown significant performance drops when models trained on simulated falls are tested on real-world falls. Several factors involved in real falls are difficult to simulate accurately. Hence, simulating falls involves assumptions that cannot be determined in advance in real-world falls. For instance, while real-world falls may lead to injuries, simu-

lated falls use controlled fall heights to prevent injuries, resulting in lower impact velocities [3]. Additionally, for ethical reasons, simulated datasets are typically created with healthy young adults whose postural control differs from that of the elderly.

In this paper, we present a simple, yet efficient approach to fall detection using state-of-the-art time series techniques, trained and tested on real-world falls. Our main contributions are:

1. We perform realistic segmentation of falls and activities of daily living in a manner that simulates real-time signals and considers the context of falls.
2. We perform fall detection using both classic tabular machine learning methods and recent state-of-the-art time series classification algorithms.
3. We evaluate the transferability of the proposed algorithms from simulated datasets to real-world falls.

Our approach requires no feature engineering and has very short inference times. We evaluate it on the FARSEEING dataset [20], a large real-world dataset with 92 fallers (mean age 76.1 ± 12.6 years) and 208 verified falls captured using inertial sensors. We also evaluate our approach on the simulated FallAllD [43] and SisFall [49] datasets, which use waist-worn sensors. For all datasets, we simulate a real-time scenario by using a fixed overlapping sliding window segmentation technique to extract falls and ADLs, followed by fall detection using state-of-the-art time series classifiers. Our methods obtain cross-validated F_1 scores up to 90.7% on FARSEEING and 97.2% on the simulated datasets. We also identify important segments in motion signals for fall detection using post-hoc explanation techniques. Our work proposes a realistic, accurate, and efficient framework for timely fall detection with great potential for real-world implementation. We achieve low daily false alarm and false negative rates of 0.29 and 0.31 respectively on real-world falls (FARSEEING), each equating to about 1 error in every 3 days. To support this paper, we have made all our code available[1].

The rest of this paper is organised as follows. Section 2 reviews related work on fall detection using time series and real-world falls. In Sect. 3, we discuss the datasets and preprocessing steps, including segmentation. Details and results of our experiments are given in Sect. 4, and Sect. 5 offers concluding remarks.

2 Related Work

In this section, we discuss recent time series approaches to fall detection, as well as fall detection algorithms evaluated on real-world falls.

2.1 Fall Detection Using Time Series Approaches

Although signals obtained from inertial sensors can be easily modelled as time series to preserve the temporal information and context of falls, most fall detection approaches have traditionally relied on manual feature extraction or deep

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

representation learning. However, a few recent approaches have modelled fall detection as a time series classification problem.

Recently, a recurrent neural network approach was proposed for fall detection, achieving recall of up to 96% on the SisFall dataset. Similarly, a long-short-term memory (LSTM) model trained and tested on the SisFall dataset achieved 96.4% recall. Another approach, the Temporal Convolutional Network and Temporal Attention Network (TCN-TAM) [35], extracted features from motion signals as time series and obtained an 88% recall on the FallAllD dataset. Likewise, the DeepFall model [18], a deep residual network, achieved an F_1 score of 92.79% on the FallAllD dataset. However, despite modelling fall signals as time series, these techniques used complex deep learning models for representation learning and classification. Furthermore, training and testing were conducted on simulated datasets, with no analysis of their transferability to real-world data.

2.2 Real-World Fall Detection

Falls are accidental and diverse, making them rare compared to activities of daily living [19]. Consequently, most fall detection research relies on simulated falls, which, despite their limitations [48], are useful for developing detection solutions. The scarcity of real-world fall datasets means that few studies have trained and evaluated techniques on actual fall data.

In a recent study [33], an ensemble of random forests was trained on a clinical dataset of 25 multiple sclerosis patients monitored over 2 months in free-living conditions, although the dataset is not publicly available for analysis. Another study [38] trained several machine learning models on the FARSEEING dataset using five manually extracted features, achieving an F_1 score of 65%. A more recent study [41] achieved an F_1 score of 91% in fall detection using a Residual Network, though the authors used only a subset (22 subjects) of FARSEEING.

3 Materials and Methods

3.1 Datasets

We evaluate our fall detection techniques using the FARSEEING dataset [20], a large collection of real-world falls. For completeness, we also evaluate our techniques on two simulated fall datasets, FallAllD [43] and SisFall [49]. SisFall uses only waist-worn sensors because signals obtained from sensors placed around the centre of mass are preferred for fall detection [37]. Therefore, we use accelerometer signals from sensors worn around the centre of mass across all datasets for consistency. Figure 1 shows sample acceleration signals from each dataset. Table 1 provides summary statistics for all the datasets after preprocessing.

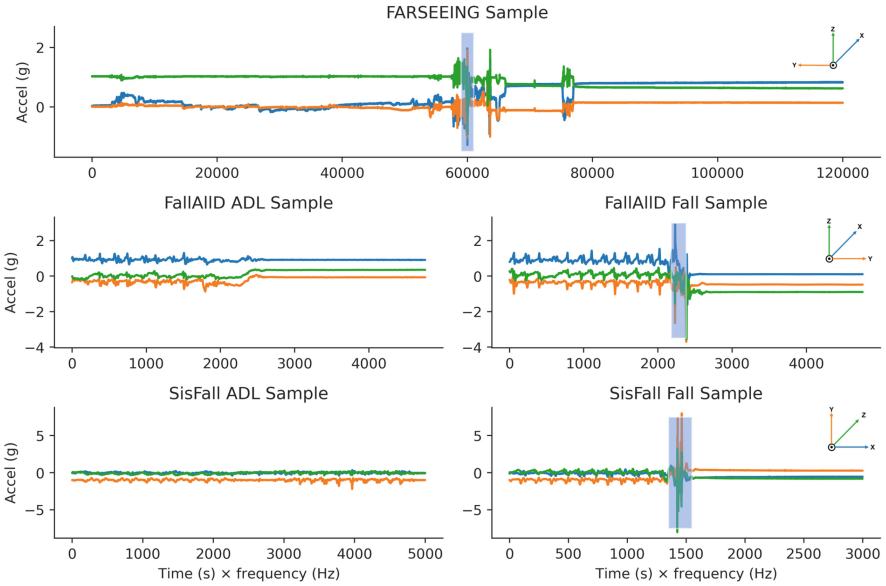


Fig. 1. Accelerometer signals from the FARSEEING, FallAllD, and SisFall datasets with the impact sample highlighted (from 1 s before to one second after the fall). The datasets have different sensor orientations; we aggregate the channels using the magnitude of acceleration, so the data becomes invariant to sensor orientation.

FARSEEING. The FARSEEING dataset [20] comprises 208 verified falls from 92 individuals (mean age 76.1 ± 12.6 years) captured using tri-axial inertial sensors. Each fall includes 20 min of data centred around the impact event, with 10 min before and after the fall. The fall signals were captured across different studies, and hence, have various sensor configurations: 72 falls with accelerometer signals only, 15 with accelerometer and gyroscope signals, and 121 with accelerometer, gyroscope, and magnetometer signals. Sensors were placed on the fifth lumbar (L5) for 150 falls and on the thigh for 58 falls. Sampling rates were 100 Hz for 152 falls and 20 Hz for 56 falls. In this study, we use accelerometer signals from 41 participants with L5 sensors, upsampling 20 Hz signals to 100 Hz using Fast Fourier Transform resampling [13] to minimise distortion. Some of the accelerometers had a range of ±2 g, while others had a range of ±6 g. Therefore, we clipped all the signals to ±2 g for uniformity.

FallAllD. The FallAllD dataset [43] is an open-source simulated falls dataset. It contains 35 types of falls and 44 types of activities of daily living (ADLs) performed by 15 participants. The dataset comprises 26,420 files collected from tri-axial accelerometers, gyroscopes, magnetometers, and barometers worn on the waist, wrist, and neck. The accelerometers have a range of ±8 g and sampling frequency of 238 Hz. In this study, we use only the accelerometer signals collected from 14 participants using waist-worn sensors.

SisFall. The SisFall dataset [49] is an open-source simulated falls dataset with simulated falls and ADLs performed by 38 healthy participants, including 23 young adults and 15 older adults. Accelerometer and gyroscope signals were collected using a waist-worn sensor sampling at 200 Hz.

3.2 Data Preprocessing

Transformation. To account for varying sensor orientations across datasets, we first aggregated the multivariate acceleration signals into orientation-invariant univariate acceleration magnitude signals: $M = \sqrt{A_x^2 + A_y^2 + A_z^2}$, where A_x, A_y, and A_z are the x, y, and z accelerations. We then performed segmentation to obtain samples, followed by standardisation across all samples to achieve zero mean and unit variance for modelling.

Segmentation. Research on falls often focuses on the immediate event, whereas the lived experience of older adults and healthcare providers considers the surrounding factors before and after the fall [48]. To capture the full context of falls and not just the "impact" event, we use a three-phase modification of the multi-phase fall model [1] with a falling phase, an impact phase, and a post-fall phase. As shown in Fig. 2, we define $[t_0, t_1)$ as the falling phase, $[t_1, t_2)$ as the impact phase, and $[t_2, t_3)$, the post-fall phase (which combines the resting and recovery phases of the original model). Each sample is $t_3 - t_0$ seconds long.

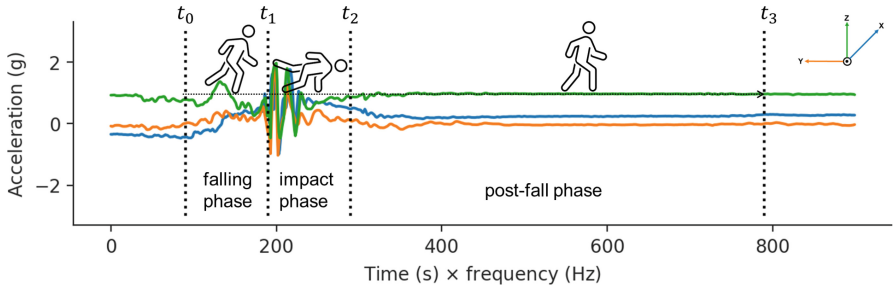


Fig. 2. Multiphase fall model on a FARSEEING fall sample.

As proposed in [38], we set the falling phase and impact phase to 1 s each. Since fall samples in SisFall are only 15 s long, we set a post-fall phase of 5 s, yielding a total sample window size of 7 s. However, using FARSEEING, we empirically demonstrate that longer post-fall phases do not always improve performance (see Appendix B.2). To simulate real-time signals, we use a fixed overlapping sliding window technique to segment samples from the accelerometer signals with a step size of 1 s (≈86% overlap).

Fall Signals Segmentation. Each fall sample in FARSEEING has 20 min of accelerometer data with 10 min before t_1 and 10 min after t_1. Hence, for FARSEEING, we first separated the portion $[t_0, t_3)$ of the signal that includes the reported fall by setting t_0 at 1 s before the impact event. We then discarded

the remainder of the post-fall signal, since the subject is likely to walk carefully after a fall [38] and the signal may not be characteristic of ADLs. ADL samples were then extracted from the 9-min signal before the impact event.

In FallAllD and SisFall, falls and ADL were recorded as separate signals. Each FallAllD signal is 20 s long with the impact event centered (at the 10th second) in the fall trial. However, SisFall fall signals are 15 s long with the impact event at arbitrary locations in the signal. For all datasets, we extracted one fall sample from each fall signal, while we used the windowing technique to extract several ADL samples from ADL signals.

Segmentation of Activities of Daily Living. Following the pattern of the multi-phase model, each ADL signal has a duration of $t_3 - t_0$ seconds. Using the phase $[t_1, t_2)$ corresponding to the impact phase as the main search window, we scan from the beginning of the sample with a step size of 1 s. For each step, we select the current $[t_0, t_3)$ as a sample if the maximum magnitude of the acceleration within the impact sample is above a threshold τ. Similar works have selected τ as high as 1.9 g [5], but we set τ at a lower value of 1.4 g to account for less energetic ADLs among older adults in the datasets.

4 Experiments

This section details baseline experiments using tabular classification models (Sect. 4.2), fall detection with time series classifiers (Sect. 4.3), cross-dataset evaluation (Sect. 4.4), and comparisons with related works (Sect. 4.5). Model explanation results are preliminary and presented in Appendix A.2. Detailed experimental results are in Appendix C.

All experiments were conducted using Python 3.11.9 on a Linux server with Ubuntu 22.04.3 LTS (1.5 TB RAM, 24 GB NVIDIA GeForce RTX 4090 GPU). We used time series classifiers from aeon [30] *v0.8.1* and tabular classifiers from scikit-learn [39] *v1.4.2*.

4.1 Evaluation Approach

Due to the small number of falls compared to ADLs (Table 1), we use a five-fold-subject cross-validation approach for evaluation. First, we shuffle and divide all subjects into five groups. Then, we use samples from the subjects in each group for testing while training on the remaining groups. We report metrics as mean $\pm$ standard deviation across all folds. We focus on AUC, Precision, Recall, and F_1 score for falls. Although adjusting classifier thresholds can increase recall by minimising undetected falls, it also reduces precision, leading to more false alarms. Therefore, we prioritise the F_1 score to balance recall and precision.

4.2 Baseline Experiments with Tabular Models

As recommended in [11], we first perform baseline experiments using tabular models. We use the scikit-learn implementations of five tabular classifiers, namely, *ExtraTrees* with 150 estimators, *K-Nearest Neighbours* (KNN, $k = 5$), *LogisticRegressionCV* ($CV = 5$), *RandomForest* with 150 estimators,

Table 1. Dataset sizes after preprocessing and segmentation

Dataset	Subjects	ADL Samples	Falls Samples	Total	Falls:ADL Ratio
FARSEEING	41	1064	145	1209	0.14
FallAllD	14	1279	466	1745	0.36
SisFall	38	10843	1200	12043	0.11

and *RidgeCV* ($CV = 5$). We perform no feature extraction but model each sample as a 1D vector $\mathbf{v} \in \mathbb{R}^L$, where $L = w \times f$, w is the window size in seconds, and f is the frequency of data capture for the target dataset. Therefore, for each dataset, we obtain $\{X \in \mathbb{R}^{N \times L}, y \in \{0, 1\}^N\}$, where X are the sample signals, y are the targets, N is the number of samples, and L is the total length of each sample. We evaluate each dataset on all the tabular models using a five-fold-subject cross-validation. Figure 3 shows boxplots of F_1 scores across the five folds for each tabular model (detailed results in Tables 7, 8, and 9 in Appendix C).

Table 2. Tabular Models Average Cross-validation Results

Dataset	Model	AUC	Precision (%)	Recall (%)	F_1 Score (%)
FARSEEING	**ExtraTrees**	99.2 ± 0.8	91.3 ± 11.3	82.6 ± 11.9	**85.9 ± 8.7**
	KNN	79.2 ± 8.8	77.1 ± 26.4	37.9 ± 13.3	47.5 ± 11.3
	LogisticCV	94.0 ± 3.7	87.2 ± 10.7	76.9 ± 14.4	81.3 ± 11.0
	RandomForest	99.0 ± 1.0	88.6 ± 12.0	83.4 ± 10.5	85.0 ± 6.7
	RidgeCV	96.0 ± 2.9	98.8 ± 2.8	67.5 ± 10.3	79.7 ± 7.5
FallAllD	**ExtraTrees**	98.8 ± 0.8	90.1 ± 7.7	92.9 ± 4.9	**91.2 ± 4.0**
	KNN	81.4 ± 7.5	84.5 ± 13.8	31.7 ± 12.0	44.6 ± 13.6
	LogisticCV	98.2 ± 0.8	89.2 ± 11.2	87.8 ± 11.4	87.6 ± 6.9
	RandomForest	98.6 ± 0.6	89.2 ± 8.9	89.7 ± 7.8	89.0 ± 5.2
	RidgeCV	98.4 ± 0.9	91.6 ± 8.6	82.9 ± 13.5	86.1 ± 7.7
SisFall	ExtraTrees	98.8 ± 0.8	91.1 ± 5.5	72.6 ± 8.5	80.4 ± 4.0
	KNN	95.2 ± 2.6	85.0 ± 4.9	77.4 ± 7.5	**80.9 ± 5.4**
	LogisticCV	75.4 ± 4.5	53.5 ± 27.1	12.3 ± 7.5	17.6 ± 10.3
	RandomForest	97.6 ± 1.1	91.2 ± 4.5	67.9 ± 8.0	77.5 ± 4.7
	RidgeCV	73.8 ± 3.5	94.3 ± 12.8	1.2 ± 1.1	2.4 ± 2.2

The best performing model and best F_1 score for each dataset is shown in **bold**

Table 2 shows the mean cross-validation results achieved by tabular models on all datasets. *ExtraTrees* achieves the best result overall on FARSEEING and FallAllD, while *KNN* achieves the best performance on SisFall. Overall, the SisFall dataset is the most challenging for the tabular classifiers. This may be a result of the arbitrary positioning of the impact event in the SisFall fall samples.

4.3 Experiments with Time Series Models

We use a univariate time series classification approach, aggregating the multivariate x, y, z channels into univariate magnitude. Initial experiments showed that

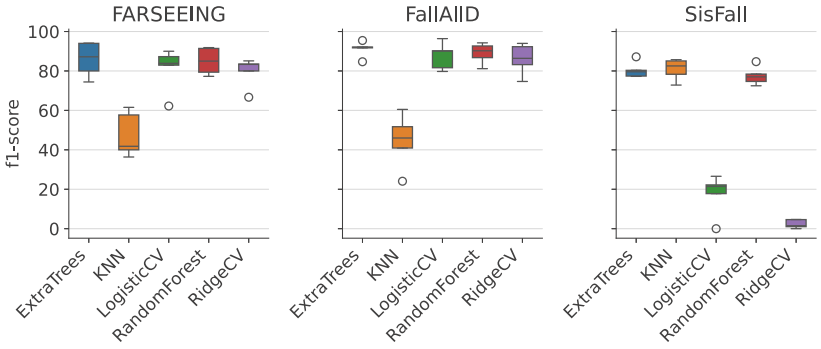


Fig. 3. Tabular Models Cross-validated F_1 scores.

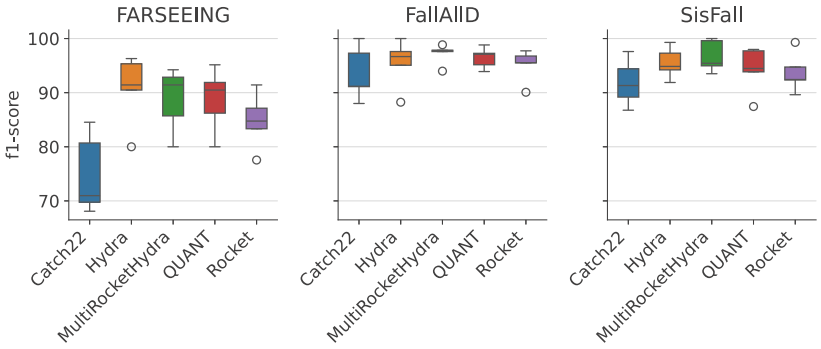


Fig. 4. Time Series Models Cross-validated F_1 scores.

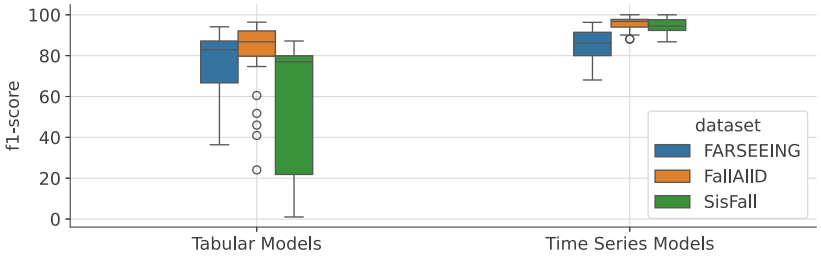


Fig. 5. Summary of all Cross-validated F_1 scores across all datasets.

using the univariate magnitude time series leads to significantly better results than using the x, y, z channels as a multivariate time series (see Appendix B.1). After reshaping each dataset as $\{X \in \mathbb{R}^{N\times 1\times L}, y \in \{0,1\}^N\}$, where X, y, N, and L are as previously defined, we performed fall detection using the *aeon toolkit* implementations of five state-of-the-art time series classifiers: *Hydra* [9], *Rocket* [8], *MultiRocketHydra*, *Catch22* [26], and *Quant* [10]. Initial single train/test split experiments with deep learning models (*InceptionTime* [17] and *FCN* [55])

Table 3. Time Series Models Average Cross-validation Results

Dataset	Model	AUC	Precision (%)	Recall (%)	F_1 Score (%)
FARSEEING	Catch22	96.6 ± 1.8	79.2 ± 10.4	72.0 ± 10.5	74.8 ± 7.3
	Hydra	94.0 ± 5.2	93.2 ± 3.9	89.1 ± 11.5	**90.7 ± 6.5**
	M.R.Hydra	93.0 ± 5.0	91.3 ± 5.8	87.4 ± 10.5	88.9 ± 5.9
	QUANT	99.4 ± 0.9	89.8 ± 6.8	88.7 ± 11.0	88.8 ± 5.9
	Rocket	90.4 ± 4.04	88.1 ± 7.26	82.5 ± 8.5	84.8 ± 5.1
FallAllD	Catch22	99.0 ± 1.0	92.2 ± 6.5	95.1 ± 5.2	93.5 ± 5.0
	Hydra	97.0 ± 3.0	95.4 ± 5.8	95.7 ± 4.2	95.5 ± 4.4
	M.R.Hydra	98.4 ± 0.6	95.9 ± 4.2	98.7 ± 1.8	**97.2 ± 1.9**
	QUANT	100.0 ± 0.0	95.9 ± 4.1	97.2 ± 2.5	96.5 ± 1.9
	Rocket	96.4 ± 2.2	96.2 ± 3.5	94.2 ± 4.6	95.1 ± 3.0
SisFall	Catch22	99.4 ± 0.6	94.5 ± 6.4	89.7 ± 6.0	91.9 ± 4.3
	Hydra	97.0 ± 2.1	96.7 ± 1.8	94.4 ± 4.9	95.5 ± 2.7
	M.R.Hydra	97.4 ± 2.4	99.2 ± 1.4	94.4 ± 4.8	**96.7 ± 2.9**
	QUANT	99.8 ± 0.5	95.6 ± 5.0	93.1 ± 4.1	94.3 ± 4.3
	Rocket	96.0 ± 2.8	95.5 ± 5.3	92.3 ± 6.2	93.7 ± 3.6

M.R.Hydra: MultiRocketHydra. The best-performing model and best F_1 score for each dataset is shown in **bold**.

showed longer training times and poorer performance compared to other models. For example, FCN achieved an F_1 score of just 14%, taking 252 s, while Hydra achieved an F_1 score of 90% in just 49 s. InceptionTime's training was stopped after over 3 h, which we deemed too long for real-time performance. Prioritising accuracy, efficiency, and real-time performance, we excluded FCN and InceptionTime from cross-validation experiments.

Table 3 shows that MultiRocketHydra achieves the best results on FallAllD and SisFall, while Hydra performs best on FARSEEING. Quant also performs well and is the fastest among the compared methods. Figure 5 demonstrates that time series models exhibit better and more consistent performance across individual datasets, with F_1 scores ranging from 68% - 100% (Fig. 4). This superior performance is expected, as these time series models are robust to variations in the positioning of the impact phase. Unlike tabular models, which treat each time step as an individual feature, time series models effectively capture the temporal dynamics and context of the signals. Detailed results for the time series models are shown in Tables 10, 11, and 12 in Appendix C.

4.4 Cross-Dataset Evaluation

We now investigate the transferability of time series models from simulated datasets to a real-world dataset by cross-evaluating each simulated dataset with

FARSEEING. To align with FARSEEING's sensor specifications—100 Hz sampling frequency and accelerometer range of ±2 g—we resampled FallAllD and SisFall to 100 Hz and clipped their acceleration values accordingly. Standardisation was applied to all datasets.

In this experiment, we employ a single subject-wise train/test split for each dataset, reserving 33% (13 subjects) of the FARSEEING dataset for testing. We use this single test set $\mathcal{T}$ of 13 subjects exclusively from FARSEEING. We explored six training scenarios: training solely on the FARSEEING training set after excluding $\mathcal{T}$, using each of the FallAllD and SisFall training sets individually, individual combinations of the FallAllD and SisFall training sets with FARSEEING (FallAllD+ and SisFall+), and a combination of FARSEEING, FallAllD, and SisFall training sets (All). Each training set—FARSEEING, FallAllD, SisFall, FallAllD+, SisFall+, and All—was evaluated on the same test set $\mathcal{T}$ for a consistent and fair comparison (detailed results in Table 13, Appendix C).

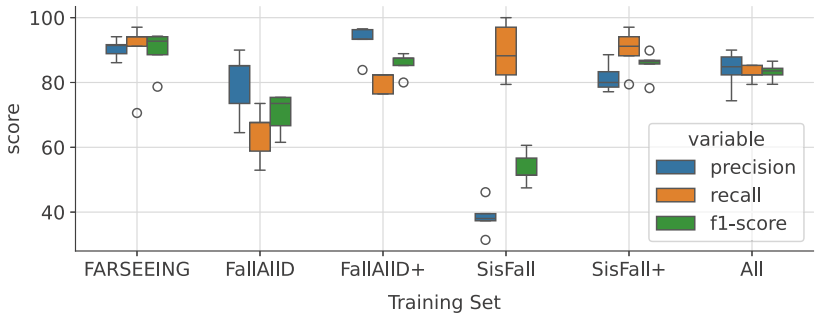


Fig. 6. Cross-dataset precision and recall obtained using time series techniques.

Figure 6 illustrates that training on FallAllD and testing on FARSEEING generally led to low recall and F_1 scores but slightly higher precision. Conversely, training on SisFall and testing on FARSEEING resulted in low precision and F_1 scores, but higher recall. This disparity indicates that training on simulated datasets and testing on real-world data can cause imbalances between precision and recall, leading to either increased missed falls or increased false alarms. Combining simulated and real-world datasets (FallAllD+ and SisFall+) improved scores, with the combination of all three training sets giving a better balance between precision and recall and lower variance overall. This suggests a promising approach of augmenting real-world fall datasets with multiple simulated datasets.

4.5 Comparison with Related Work

We compare our best results with results obtained with other techniques evaluated on FARSEEING, FallAllD, and SisFall. On FARSEEING, we compare

with [38], which uses manual feature extraction and ResNet [41] evaluated on 22 subjects. On FallAllD, comparisons include TCN-TAM [35], DeepFall [18], and CNN-GRU [23] employing convolutional neural networks and gated recurrent units. On SisFall, comparisons are made with LSTM-CVAE [54], an RNN-based approach [34], and an LSTM-based method [28].

We report each study's cross-validation approach, recall, and F_1 score as documented by the respective authors (Table 4). Despite employing simple and efficient time series techniques without explicit feature extraction, our results are competitive with those achieved using deep learning models. However, this comparison is only indicative due to differences in preprocessing steps.

Table 4. Performance Comparison with Related Work

Method	Dataset	CV	Recall (%)	F_1 Score (%)
Feature-based [38]	FARSEEING	5-fold subject CV	81.1	64.6
ResNet* [41]		NR	94.0	91.0
Ours (proposed)		5-fold subject CV	89.1 ± 11.5	90.7 ± 6.5
TCN-TAM [35]	FallAllD	NR	88.0	89.0
DeepFall [18]		5-fold	NR	92.8
CNN and GRU [23]		LOSO	92.5	94.3
Ours (proposed)		5-fold subject CV	98.7 ± 1.8	97.2 ± 1.9
LSTM-CVAE [54]	SisFall	NR	99.0	95.0
RNN [34]		NR	96.7	NR
LSTM [28]		5-fold	96.4	NR
Ours (proposed)		5-fold subject CV	94.4 ± 4.8	96.7 ± 2.9

*This approach was trained and tested on only a subset of the dataset (22 subjects). **NR**: Not reported. **LOSO**: Leave-One-Subject-Out cross-validation

As shown in Table 5 (Appendix A.1), the *Hydra* model, which performs best on average on the FARSEEING dataset, achieves a false alarm rate of 0.012 per hour and a miss rate of 0.013 per hour. This corresponds to ≈28% chance of one false alarm and a 31% chance of one missed fall per day. Therefore, a deployed Hydra model could have as few as 1 false alarm and 1 missed fall in 3 days.

5 Conclusion

In this work, we presented a framework for real-world fall detection using state-of-the-art time series techniques applied to real-world fall data. Our approach is simple, efficient, and requires no manual feature engineering. We performed segmentation in a manner that imitates real-time signals and includes the pre-fall and post-fall contexts, with about 1 false alarm and 1 false negative in 3 days. While our primary dataset was real-world, we also evaluated our

method on two simulated datasets and achieved comparable results. Experimentation with both tabular and time series models across all datasets consistently demonstrated superior performance of time series models. We also performed cross-dataset evaluation and found significant reductions in precision and recall when models trained on simulated data were applied to real-world falls. However, we found that extending real-world falls with multiple simulated datasets is a promising approach. Finally, we used a post-hoc explanation technique to highlight segments of the motion time series that are relevant for each classifier (Appendix A.2).

We hereby propose four main practical considerations for fall detection, namely, *reality*, *accuracy*, *timeliness*, and *explainability* (**RATE**). A *realistic* fall detection algorithm should be validated, at least partially, using real-world falls, possibly through a blend of simulated and real-world data. For *accuracy*, algorithms must strike a balance between precision and recall, considering that missed falls and false alarms are both undesirable. Real-time detection (*timeliness*) is crucial to prevent prolonged immobility after a fall, known as "long-lie", which can lead to severe consequences. Finally, we suggest incorporating *explanations* and online learning mechanisms to refine models based on segments that contribute to false alarms and false negatives.

Although our proposed fall detection framework has good **RATE** characteristics, there are a few limitations that will be addressed in future work. Currently, we use a simple univariate time series approach for compatibility with tabular baseline models. Future work will explore a multivariate approach using all accelerometer axes and additional signals from the magnetometer and gyroscope sensors. Deep learning models were excluded for efficiency reasons, but we plan to investigate recent efficient models like LITE [16] and explore the use of reasonable contract times for training models (see [12]). Lastly, the scarcity of publicly available real-world fall datasets is a challenge. Future efforts will focus on curating and releasing open-source real-world fall datasets to advance fall detection research.

Acknowledgments. This study has received funding from Science Foundation Ireland [12/RC/2289_P2] at the Insight SFI Research Centre for Data Analytics and the European Union's H2020 Marie Skłodowska-Curie Cofund programme, NeuroInsight [Grant ID: 101034252].

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A Usability and Explainability

Appendix A.1 Estimated False Alarm and Miss Rates

Since we use an overlapping sliding window with a step size of 1 s, each classifier would process a total of $1 \times 60(\text{s}) \times 60(\text{min}) = 3600$ samples per hour in practice.

Taking the *Hydra* classifier as an example, we obtain a rough estimate of false alarm and miss rates per hour (Table 5) in the following manner.

Let P be the total number of falls, and N be the total number of ADLs per hour, such that $P+N = 3600$. From the Fall: ADL ratio in Table 1, we know that

$$P = 0.14N \tag{1}$$

So that

$$0.14N + N = 3600 \tag{2}$$

Hence, we can estimate the number of ADLs per hour as $N = \frac{3600}{1.14} \approx 3158$, and the number of falls $P = 3600 - 3158 = 442$ per hour. Therefore, we estimate the number of misses per hour, FN as

$$FN = P \times (1 - Recall) \tag{3}$$

$$FN = 442 \times (1 - 0.8913) \tag{4}$$

$$FN \approx 48 \tag{5}$$

Similarly, false alarms per hour, FP:

$$FP = N \times (1 - Specificity) \tag{6}$$

$$FP = 3158 \times (1 - 0.986) \tag{7}$$

$$FP \approx 44 \tag{8}$$

Hence, miss rate $= \frac{48}{3600} \approx 0.013$, and false alarm rate $= \frac{44}{3600} \approx 0.012$ per hour.

Table 5. False Alarm Rate and Miss Rate Per Hour

Model	Ave. Recall	Ave. Specificity	False Alarm Rate	Miss Rate
Catch22	72.04	96.55	0.030	0.034
Hydra	89.13	98.60	0.012	0.013
MultiRocketHydra	87.37	98.31	0.015	0.016
QUANT	88.74	98.34	0.015	0.014
Rocket	82.52	98.50	0.013	0.021

Appendix A.2 Model Explanation

We use *tscaptum* [7,44] to identify time intervals in motion data that influence classifier decisions on FARSEEING. *tscaptum* groups adjacent time points into segments of size c to enhance robustness and reduce runtime, with important intervals identified by iteratively masking segments. SHAP (SHapley Additive exPlanations) scores [27] are then obtained based on their impact on predictions.

We perform a subject-wise train-test split, using 30% (13 subjects) for testing and 70% (28 subjects) for training (see Fig. 7). Temporal SHAP scores are obtained on the test set with $c = 100$, representing a 1-s interval (sampled at 100 Hz). This produces equally distributed attribution scores within each chunk, matching the shape of the input sample.

We present attribution profiles for representative true positives, false positives, true negatives, and false negatives across all classifiers. Although the impact phase occurs within $t = [1, 2)$, attribution scores for *Hydra* (Fig. 8) and *MultiRocketHydra* (Fig. 10) suggest uniform importance across all phases. However, *Rocket* (Fig. 9), *Catch22* (Fig. 11), and *QUANT* (Fig. 12) show high scores between the end of the falling phase and the start of the impact phase.

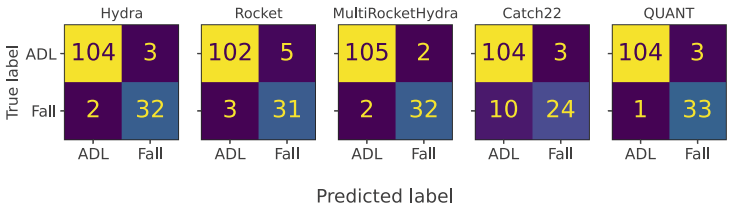


Fig. 7. Confusion matrices for time series classifiers on the FARSEEING test set.

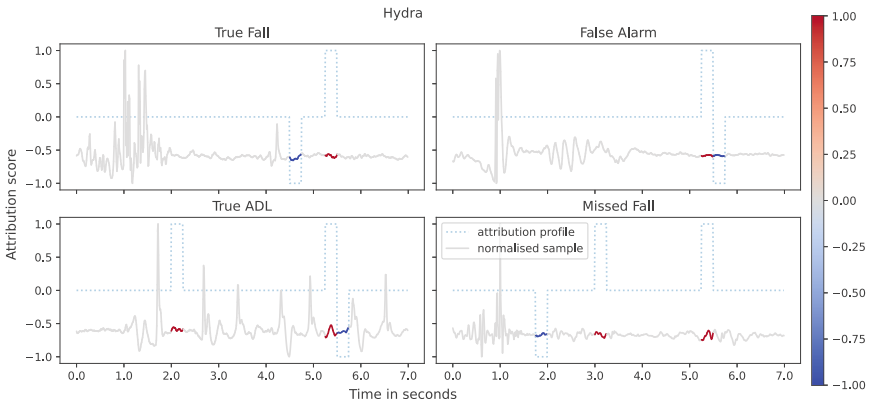


Fig. 8. Hydra + SHAP temporal attribution profiles for some samples.

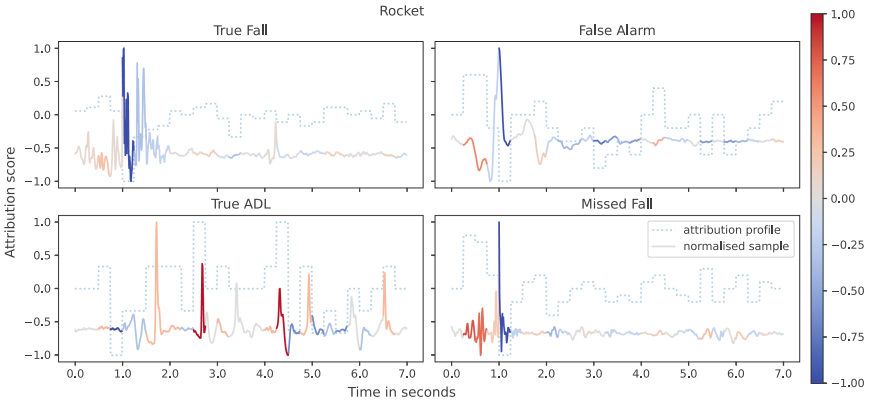


Fig. 9. Rocket + SHAP temporal attribution profiles for some samples.

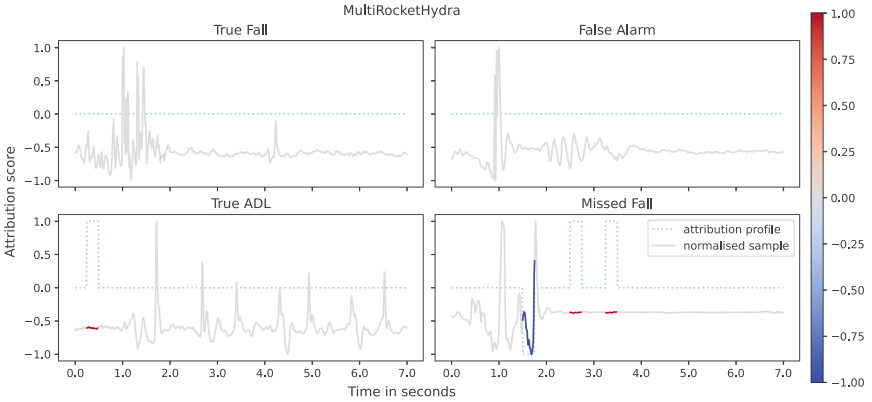


Fig. 10. MultiRocketHydra + SHAP temporal attribution profiles for some samples.

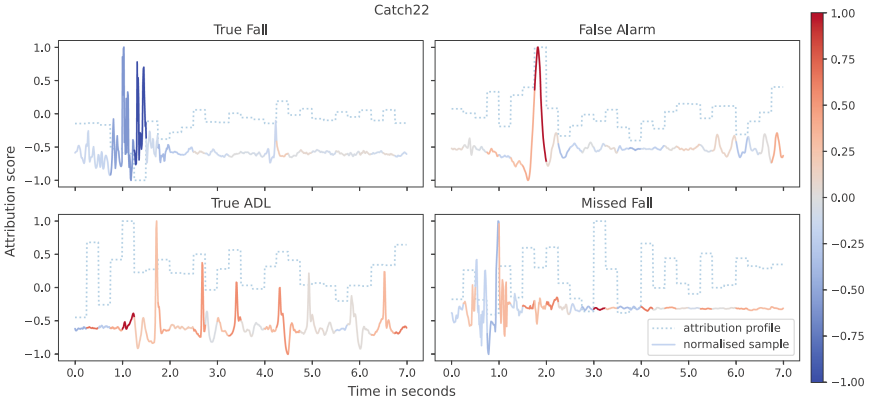


Fig. 11. Catch22 + SHAP temporal attribution profiles for some samples.

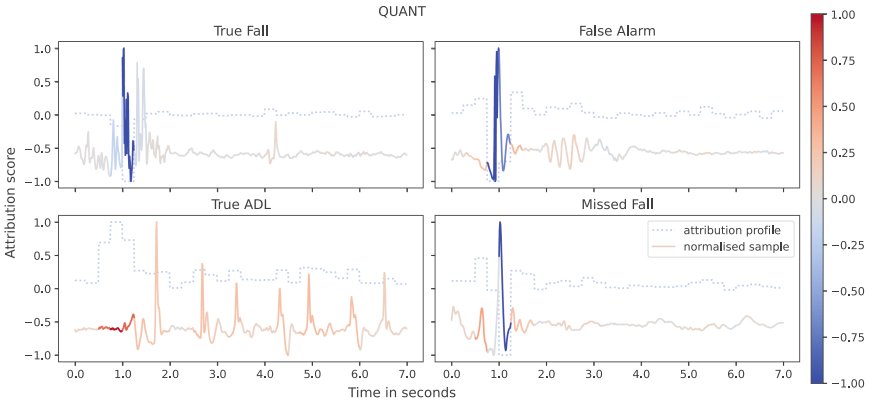


Fig. 12. QUANT + SHAP temporal attribution profiles for some samples.

Appendix A.3 Model Runtime

In our experiments, we compute runtime in milliseconds (ms) as the time it takes each model to perform inference on one sample. As shown in Fig. 13, the tabular models run extremely fast. However, the time series models also run fast enough to be deployed in real-time, with *MultiRocketHydra* having the slowest inference speed of 144 ms per sample in one instance.

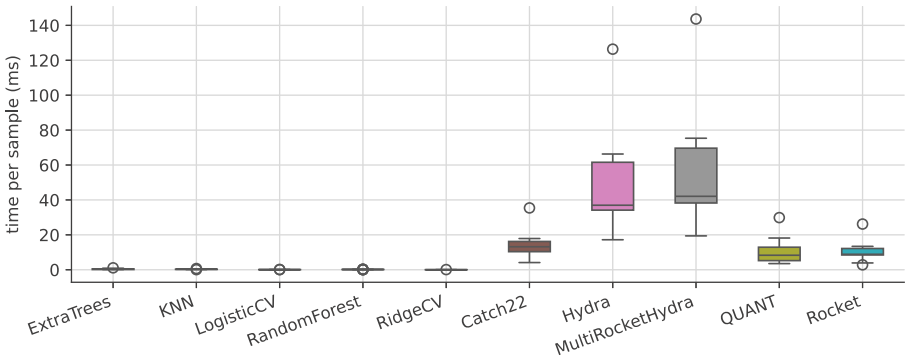


Fig. 13. Inference time per sample in milliseconds for all models.

Appendix B Preprocessing Choices

Appendix B.1 Univariate vs Multivariate

We performed a single train/test split experiment on FARSEEING to compare the performance of univariate acceleration magnitude with multivariate x, y, z acceleration signals. As shown in Fig. 14, using univariate magnitude signals with the time series classifiers showed significantly better scores in all metrics.

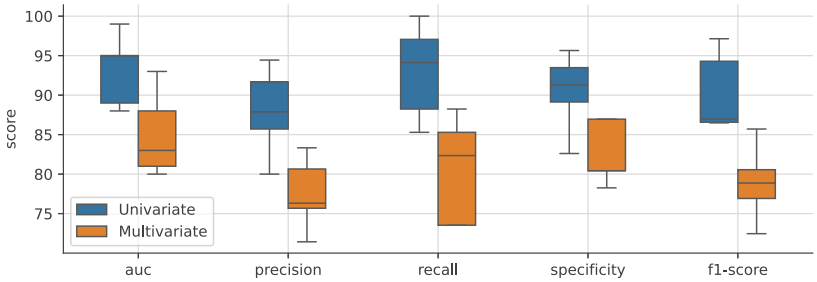


Fig. 14. Univariate acceleration magnitude vs. multivariate x, y, z acceleration on FARSEEING. Each box represents the performance of all time series classifiers.

Appendix B.2 Window Size and Post-fall Phase

We trained the time series models on the FARSEEING dataset with total sample window sizes ranging from 3 s (1-s post-fall phase) to 27 s (25-s post-fall phase). As shown in Fig. 15, longer post-fall phases improve AUC but don't necessarily improve recall, specificity, or F_1 scores. In fact, they may degrade performance and increase runtime. Experts recommend that a 5-s post-fall phase is sufficient to capture the necessary context.

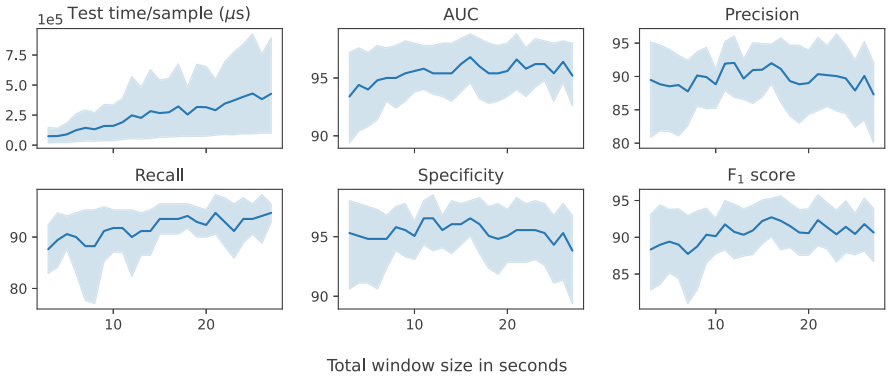


Fig. 15. Effect of post-fall window size on model performance on the FARSEEING dataset.

Appendix C Extended Cross-Validation Results

Here, we give more detailed results, including cross-validation split sizes and the performance of each classifier on individual splits. We report the inference time per sample (T) in milliseconds, AUC, precision, recall, specificity, and F_1 score.

Table 6. Cross-validation splits

Dataset	Fold	Train Set				Test Set			
		Subjects	ADLs	Falls	Fall : ADL	Subjects	ADLs	Falls	Fall : ADL
FARSEEING	0	33	865	118	0.14	8	304	27	0.09
	1	33	684	119	0.17	8	485	26	0.05
	2	33	1094	125	0.11	8	75	20	0.27
	3	33	1039	127	0.12	8	130	18	0.14
	4	33	994	92	0.09	8	175	53	0.30
FallAllD	0	12	1082	373	0.34	2	197	93	0.47
	1	12	1077	388	0.36	2	202	78	0.39
	2	12	1058	403	0.38	2	221	63	0.29
	3	12	1090	399	0.37	2	189	67	0.35
	4	12	1199	423	0.35	2	80	43	0.54
SisFall	0	31	8957	1004	0.11	7	1886	196	0.10
	1	31	8883	847	0.10	7	1960	353	0.18
	2	31	8706	993	0.11	7	2137	207	0.10
	3	31	8720	1073	0.12	7	2123	127	0.06
	4	31	8945	937	0.10	7	1898	263	0.14

Table 7. Performance of Tabular Models on the FARSEEING Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
ExtraTrees	0.40	99.0	72.7	88.9	97.0	80.0	0
	0.30	99.0	94.1	61.5	99.8	74.4	1
	1.10	98.0	89.5	85.0	97.3	87.2	2
	0.70	100.0	100.0	88.9	100.0	94.1	3
	0.50	100.0	100.0	88.7	100.0	94.0	4
KNN	0.10	80.0	60.0	55.6	96.7	57.7	0
	0.10	78.0	85.7	23.1	99.8	36.4	1
	0.30	65.0	40.0	40.0	84.0	40.0	2
	0.20	86.0	100.0	44.4	100.0	61.5	3
	0.10	87.0	100.0	26.4	100.0	41.8	4
LogisticCV	0.10	97.0	85.7	88.9	98.7	87.3	0
	0.00	88.0	73.7	53.9	99.0	62.2	1
	0.10	93.0	81.0	85.0	94.7	82.9	2
	0.10	95.0	100.0	72.2	100.0	83.9	3
	0.10	97.0	95.7	84.9	98.9	90.0	4

(*continued*)

Table 7. (*continued*)

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
RandomForest	0.10	99.0	69.4	92.6	96.4	79.4	0
	0.00	98.0	94.4	65.4	99.8	77.3	1
	0.20	98.0	85.0	85.0	96.0	85.0	2
	0.10	100.0	94.1	88.9	99.2	91.4	3
	0.10	100.0	100.0	84.9	100.0	91.8	4
RidgeCV	0.00	97.0	100.0	74.1	100.0	85.1	0
	0.00	91.0	100.0	50.0	100.0	66.7	1
	0.00	96.0	93.8	75.0	98.7	83.3	2
	0.10	98.0	100.0	66.7	100.0	80.0	3
	0.00	98.0	100.0	71.7	100.0	83.5	4

Table 8. Performance of Tabular Models on the FallAllD Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
ExtraTrees	0.51	100.0	98.8	86.0	99.5	92.0	0
	0.51	99.0	87.4	97.4	94.6	92.1	1
	0.47	98.0	78.4	92.1	92.8	84.7	2
	0.50	98.0	92.4	91.0	97.4	91.7	3
	0.87	99.0	93.3	97.7	96.3	95.5	4
KNN	0.25	76.0	86.7	14.0	99.0	24.1	0
	0.2	88.0	87.8	46.2	97.5	60.5	1
	0.25	75.0	62.2	36.5	93.7	46.0	2
	0.27	77.0	85.7	26.9	98.4	40.9	3
	0.69	91.0	100.0	34.9	100.0	51.7	4
LogisticCV	0.08	99.0	96.9	67.7	99.0	79.8	0
	0.08	98.0	86.9	93.6	94.6	90.1	1
	0.08	97.0	71.4	95.2	89.1	81.6	2
	0.08	98.0	90.9	89.6	96.8	90.2	3
	0.17	99.0	100.0	93.0	100.0	96.4	4
RandomForest	0.15	99.0	98.6	77.4	99.5	86.8	0
	0.16	99.0	88.4	97.4	95.1	92.7	1
	0.16	98.0	74.7	88.9	91.4	81.2	2
	0.17	98.0	90.9	89.6	96.8	90.2	3
	0.35	99.0	93.2	95.4	96.3	94.3	4
RidgeCV	0.03	99.0	98.3	60.2	99.5	74.7	0
	0.03	99.0	92.3	92.3	97.0	92.3	1
	0.03	97.0	77.0	90.5	92.3	83.2	2
	0.03	98.0	93.1	80.6	97.9	86.4	3
	0.04	99.0	97.5	90.7	98.8	94.0	4

Table 9. Performance of Tabular Models on the SisFall Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
ExtraTrees	0.14	99.0	99.2	63.3	100.0	77.3	0
	0.12	99.0	89.8	84.7	98.3	87.2	1
	0.13	98.0	86.9	73.4	98.9	79.6	2
	0.13	100.0	85.7	75.6	99.3	80.3	3
	0.13	98.0	94.0	65.8	99.4	77.4	4
KNN	0.31	96.0	90.3	76.0	99.2	82.6	0
	0.28	98.0	87.9	82.4	98.0	85.1	1
	0.29	92.0	85.2	72.5	98.8	78.3	2
	0.30	97.0	84.1	87.4	99.0	85.7	3
	0.31	93.0	77.4	68.8	97.2	72.8	4
LogisticCV	0.07	70.0	33.3	15.8	96.7	21.5	0
	0.07	82.0	100.0	0.0	100.0	0.0	1
	0.07	77.0	42.6	19.3	97.5	26.6	2
	0.07	75.0	37.7	15.8	98.5	22.2	3
	0.07	73.0	53.9	10.7	98.7	17.8	4
RandomForest	0.17	98.0	98.3	60.2	99.9	74.7	0
	0.15	98.0	90.9	79.3	98.6	84.7	1
	0.15	97.0	86.2	69.6	98.9	77.0	2
	0.16	99.0	89.0	70.1	99.5	78.4	3
	0.15	96.0	91.3	60.1	99.2	72.5	4
RidgeCV	0.02	72.0	100.0	0.5	100.0	1.0	0
	0.02	78.0	100.0	0.0	100.0	0.0	1
	0.02	77.0	71.4	2.4	99.9	4.7	2
	0.02	70.0	100.0	2.4	100.0	4.6	3
	0.02	72.0	100.0	0.8	100.0	1.5	4

Table 10. Performance of Time Series Models on the FARSEEING Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
Catch22	5.8	96.0	76.7	85.2	97.7	80.7	0
	4.1	95.0	76.2	61.5	99.0	68.1	1
	17.9	95.0	65.2	75.0	89.3	69.8	2
	9.3	99.0	84.6	61.1	98.5	71.0	3
	7.6	98.0	93.2	77.4	98.3	84.5	4
Hydra	18.2	98.0	96.3	96.3	99.7	96.3	0
	17.2	85.0	94.7	69.2	99.8	80.0	1
	61.4	96.0	86.4	95.0	96.0	90.5	2
	37.2	94.0	94.1	88.9	99.2	91.4	3
	30.3	97.0	94.4	96.2	98.3	95.3	4

(*continued*)

Table 10. (*continued*)

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
MultiRocketHydra	25.1	98.0	89.7	96.3	99.0	92.9	0
	19.4	85.0	94.7	69.2	99.8	80.0	1
	69.7	92.0	81.8	90.0	94.7	85.7	2
	44.9	94.0	94.1	88.9	99.2	91.4	3
	33.3	96.0	96.1	92.5	98.9	94.2	4
QUANT	5.4	100.0	80.7	92.6	98.0	86.2	0
	7.9	99.0	94.7	69.2	99.8	80.0	1
	18.2	98.0	86.4	95.0	96.0	90.5	2
	8.3	100.0	89.5	94.4	98.5	91.9	3
	10.5	100.0	98.0	92.5	99.4	95.2	4
Rocket	3.9	95.0	78.1	92.6	97.7	84.8	0
	2.8	86.0	82.6	73.1	99.2	77.6	1
	12.6	87.0	93.8	75.0	98.7	83.3	2
	8.4	94.0	94.1	88.9	99.2	91.4	3
	5.9	90.0	91.7	83.0	97.7	87.1	4

Table 11. Performance of Time Series Models on the FallAllD Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
Catch22	15.9	100.0	97.8	96.8	99.0	97.3	0
	15.5	99.0	84.6	98.7	93.1	91.1	1
	16.4	98.0	88.7	87.3	96.8	88.0	2
	16.0	98.0	89.9	92.5	96.3	91.2	3
	35.4	100.0	100.0	100.0	100.0	100.0	4
Hydra	63.8	97.0	100.0	93.6	100.0	96.7	0
	61.7	98.0	91.7	98.7	96.5	95.1	1
	60.1	98.0	98.4	96.8	99.6	97.6	2
	66.3	92.0	87.0	89.6	95.2	88.2	3
	126.4	100.0	100.0	100.0	100.0	100.0	4
MultiRocketHydra	67.9	98.0	98.9	96.8	99.5	97.8	0
	75.3	98.0	88.6	100.0	95.1	94.0	1
	69.6	98.0	98.4	96.8	99.6	97.6	2
	74.6	99.0	95.7	100.0	98.4	97.8	3
	143.6	99.0	97.7	100.0	98.8	98.9	4

(*continued*)

Table 11. (*continued*)

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
QUANT	9.4	100.0	98.9	95.7	99.5	97.3	0
	13.5	100.0	89.5	98.7	95.5	93.9	1
	12.3	100.0	96.7	93.7	99.1	95.2	2
	17.3	100.0	94.4	100.0	97.9	97.1	3
	29.9	100.0	100.0	97.7	100.0	98.8	4
Rocket	11.9	96.0	100.0	91.4	100.0	95.5	0
	12.1	97.0	94.9	96.2	98.0	95.5	1
	12.2	97.0	98.4	95.2	99.6	96.8	2
	13.4	93.0	92.2	88.1	97.4	90.1	3
	26.2	99.0	95.6	100.0	97.5	97.7	4

Table 12. Performance of Time Series Models on the SisFall Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
Catch22	12.7	99.0	83.5	90.3	98.1	86.8	0
	11.5	100.0	97.5	97.7	99.5	97.6	1
	12.5	99.0	95.6	83.6	99.6	89.2	2
	16.9	100.0	95.9	92.9	99.8	94.4	3
	13.2	99.0	100.0	84.0	100.0	91.3	4
Hydra	36.6	97.0	95.8	93.9	99.6	94.9	0
	35.5	99.0	99.7	98.9	100.0	99.3	1
	32.8	94.0	96.8	87.4	99.7	91.9	2
	35.9	99.0	95.5	99.2	99.7	97.3	3
	37.0	96.0	95.7	92.8	99.4	94.2	4
MultiRocketHydra	41.3	96.0	98.9	91.3	99.9	95.0	0
	38.1	100.0	100.0	100.0	100.0	100.0	1
	38.4	95.0	96.9	90.3	99.7	93.5	2
	40.4	100.0	100.0	99.2	100.0	99.6	3
	42.1	96.0	100.0	91.3	100.0	95.4	4
QUANT	4.4	100.0	94.3	93.4	99.4	93.9	0
	4.4	100.0	97.7	97.7	99.6	97.7	1
	3.5	99.0	87.4	87.4	98.8	87.4	2
	5.1	100.0	100.0	96.1	100.0	98.0	3
	5.9	100.0	98.4	90.9	99.8	94.5	4
Rocket	9.2	95.0	88.9	90.3	98.8	89.6	0
	8.7	99.0	100.0	98.6	100.0	99.3	1
	8.5	94.0	97.8	87.4	99.8	92.4	2
	8.7	99.0	90.7	99.2	99.4	94.7	3
	8.9	93.0	100.0	85.9	100.0	92.4	4

Table 13. Results of Cross-Dataset Evaluation

Training Set	Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score
FARSEEING	Hydra	47.7	96.0	91.4	94.1	97.2	92.8
	Rocket	8.7	93.0	86.1	91.2	95.3	88.6
	MultiRocketHydra	49.9	96.0	94.1	94.1	98.1	94.1
	Catch22	18.5	97.0	88.9	70.6	97.2	78.7
	QUANT	43.4	100.0	91.7	97.1	97.2	94.3
FallAllD	Hydra	40.2	76.0	90.0	52.9	98.1	66.7
	Rocket	9.0	74.0	64.5	58.8	89.7	61.5
	MultiRocketHydra	43.3	82.0	85.2	67.7	96.3	75.4
	Catch22	13.5	86.0	73.5	73.5	91.6	73.5
	QUANT	41.5	91.0	85.2	67.7	96.3	75.4
FallAllD+	Hydra	58.9	91.0	96.6	82.4	99.1	88.9
	Rocket	13.6	86.0	83.9	76.5	95.3	80.0
	MultiRocketHydra	76.8	90.0	93.3	82.4	98.1	87.5
	Catch22	29.9	98.0	93.3	82.4	98.1	87.5
	QUANT	69.7	98.0	96.3	76.5	99.1	85.3
SisFall	Hydra	150.4	69.0	38.0	79.4	58.9	51.4
	Rocket	47.5	76.0	39.5	100.0	51.4	56.7
	MultiRocketHydra	195.0	69.0	37.3	82.4	56.1	51.4
	Catch22	93.4	88.0	31.4	97.1	32.7	47.5
	QUANT	81.7	84.0	46.2	88.2	67.3	60.6
SisFall+	Hydra	180.8	93.0	80.0	94.1	92.5	86.5
	Rocket	52.5	94.0	78.6	97.1	91.6	86.8
	MultiRocketHydra	218.7	94.0	88.6	91.2	96.3	89.9
	Catch22	109.7	95.0	77.1	79.4	92.5	78.3
	QUANT	54.9	98.0	83.3	88.2	94.4	85.7
All	Hydra	197.9	91.0	87.9	85.3	96.3	86.6
	Rocket	59.1	88.0	74.4	85.3	90.7	79.5
	MultiRocketHydra	233.9	88.0	90.0	79.4	97.2	84.4
	Catch22	141.6	94.0	84.9	82.4	95.3	83.6
	QUANT	122.8	95.0	82.4	82.4	94.4	82.4

References

1. Becker, C., et al.: Proposal for a multiphase fall model based on real-world fall recordings with body-fixed sensors. Z. Gerontol. Geriatr. **45**(8), 707 (2012)
2. Beddiar, D.R., Oussalah, M., Nini, B.: Fall detection using body geometry and human pose estimation in video sequences. J. Vis. Commun. Image Represent. **82**, 103407 (2022)
3. Borrelli, J., Creath, R.A., Rogers, M.W.: A method for simulating forward falls and controlling impact velocity. MethodsX **11**, 102399 (2023)
4. Camp, K., Murphy, S., Pate, B.: Integrating fall prevention strategies into ems services to reduce falls and associated healthcare costs for older adults. Clin. Interventions Aging 561–569 (2024)
5. Chen, J., Kwong, K., Chang, D., Luk, J., Bajcsy, R.: Wearable sensors for reliable fall detection. In: 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, pp. 3551–3554. IEEE (2006)
6. Choi, A., et al.: Deep learning-based near-fall detection algorithm for fall risk monitoring system using a single inertial measurement unit. IEEE Trans. Neural Syst. Rehabil. Eng. **30**, 2385–2394 (2022)
7. Serramazza, D.I., Le Nguyen, T., Ifrim G.: A short tutorial for multivariate time series explanation using tscaptum. Softw. Impacts (2024), in Press. https://doi.org/10.1016/j.simpa.2024.100723, https://github.com/mlgig/tscaptum
8. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Disc. **34**(5), 1454–1495 (2020)
9. Dempster, A., Schmidt, D.F., Webb, G.I.: Hydra: competing convolutional kernels for fast and accurate time series classification. Data Min. Knowl. Disc. **37**(5), 1779–1805 (2023)
10. Dempster, A., Schmidt, D.F., Webb, G.I.: Quant: a minimalist interval method for time series classification. Data Min. Knowl. Discov. 1–26 (2024)
11. Dhariyal, B., Le Nguyen, T., Ifrim, G.: Back to basics: a sanity check on modern time series classification algorithms. In: Ifrim, G., et al. (eds.) Advanced Analytics and Learning on Temporal Data, pp. 205–229. Springer, Cham (2023)
12. Flynn, M., Large, J., Bagnall, T.: The contract random interval spectral ensemble (c-RISE): the effect of contracting a classifier on accuracy. In: Pérez García, H., Sánchez González, L., Castejón Limas, M., Quintián Pardo, H., Corchado Rodríguez, E. (eds.) HAIS 2019. LNCS (LNAI), vol. 11734, pp. 381–392. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29859-3_33
13. Hawkins, W.: Fourier transform resampling: theory and application [medical imaging]. In: 1996 IEEE Nuclear Science Symposium. Conference Record, vol. 3, pp. 1491–1495 (1996)
14. He, J., Zhang, Z., Wang, X., Yang, S.: A low power fall sensing technology based on FD-CNN. IEEE Sens. J. **19**(13), 5110–5118 (2019)
15. Hu, X., Chu, L., Pei, J., Liu, W., Bian, J.: Model complexity of deep learning: a survey. Knowl. Inf. Syst. **63**, 2585–2619 (2021)
16. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Lite: light inception with boosting techniques for time series classification. In: 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10 (2023)
17. Ismail Fawaz, H., et al.: Inceptiontime: finding alexnet for time series classification. Data Min. Knowl. Disc. **34**(6), 1936–1962 (2020)

18. Jitpattanakul, A.: Wearable fall detection based on motion signals using hybrid deep residual neural network. In: Multi-disciplinary Trends in Artificial Intelligence: 15th International Conference, MIWAI 2022, Virtual Event, 17–19 November 2022, Proceedings. vol. 13651, p. 216. Springer (2022)
19. Khan, S.S., Hoey, J.: Review of fall detection techniques: a data availability perspective. Med. Eng. Phys. **39**, 12–22 (2017)
20. Klenk, J., et al.: The farseeing real-world fall repository: a large-scale collaborative database to collect and share sensor signals from real-world falls. Eur. Rev. Aging Phys. Activity **13**, 1–7 (2016)
21. La Blunda, L., Gutierrez-Madronal, L., Wagner, M.F., Medina-Bulo, I.: A wearable fall detection system based on body area networks. IEEE Access **8**, 193060–193074 (2020)
22. Le, H.L., Nguyen, D.N., Nguyen, T.H., Nguyen, H.N.: A novel feature set extraction based on accelerometer sensor data for improving the fall detection system. Electronics **11**(7), 1030 (2022)
23. Liu, C.P., et al.: Deep learning-based fall detection algorithm using ensemble model of coarse-fine CNN and GRU networks. In: 2023 IEEE International Symposium on Medical Measurements and Applications (MeMeA), pp. 1–5. IEEE (2023)
24. Liu, J., et al.: A review of wearable sensors based fall-related recognition systems. Eng. Appl. Artif. Intell. **121**, 105993 (2023)
25. Liu, K.C., Hung, K.H., Hsieh, C.Y., Huang, H.Y., Chan, C.T., Tsao, Y.: Deep-learning-based signal enhancement of low-resolution accelerometer for fall detection systems. IEEE Trans. Cogn. Dev. Syst. **14**(3), 1270–1281 (2021)
26. Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S.: catch22: canonical time-series characteristics: selected through highly comparative time-series analysis. Data Min. Knowl. Disc. **33**(6), 1821–1852 (2019)
27. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, pp. 4768–4777. Curran Associates Inc., Red Hook (2017)
28. Magalhães, C., Ribeiro, J., Leite, A., Solteiro Pires, E., Pavão, J.: Automatic fall detection using long short-term memory network. In: International Work-Conference on Artificial Neural Networks, pp. 359–371. Springer (2021)
29. Mekruksavanich, S., Jantawong, P., Hnoohom, N., Jitpattanakul, A.: Wearable fall detection based on motion signals using hybrid deep residual neural network. In: International Conference on Multi-disciplinary Trends in Artificial Intelligence, pp. 216–224. Springer (2022)
30. Middlehurst, M., et al.: aeon: a python toolkit for learning from time series. arXiv preprint arXiv:2406.14231 (2024)
31. Montero-Odasso, M., et al.: World guidelines for falls prevention and management for older adults: a global initiative. Age Ageing **51**(9), afac205 (2022)
32. Mosquera-Lopez, C., et al.: Automated detection of real-world falls: Modeled from people with multiple sclerosis. IEEE J. Biomed. Health Inform. **25**(6), 1975–1984 (2020)
33. Mosquera-Lopez, C., et al.: Automated detection of real-world falls: modeled from people with multiple sclerosis. IEEE J. Biomed. Health Inform. **25**(6), 1975–1984 (2021)
34. Musci, M., De Martini, D., Blago, N., Facchinetti, T., Piastra, M.: Online fall detection using recurrent neural networks on smart wearable devices. IEEE Trans. Emerg. Top. Comput. **9**(3), 1276–1289 (2020)

35. Nguyen, D.A., Pham, C., Argent, R., Caulfield, B., Le-Khac, N.A.: Model and empirical study on multi-tasking learning for human fall detection. Vietnam J. Comput. Sci. 1–14 (2024)
36. Nho, Y.H., Ryu, S., Kwon, D.S.: UI-GAN: generative adversarial network-based anomaly detection using user initial information for wearable devices. IEEE Sens. J. **21**(8), 9949–9958 (2021)
37. Özdemir, A.T.: An analysis on sensor locations of the human body for wearable fall detection devices: principles and practice. Sensors **16**(8), 1161 (2016)
38. Palmerini, L., Klenk, J., Becker, C., Chiari, L.: Accelerometer-based fall detection using machine learning: training and testing on real-world falls. Sensors **20**(22), 6479 (2020)
39. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
40. Pernes, M., Agostinho, I., Bernardes, R.A., Fernandes, J.B., Baixinho, C.L.: Documenting fall episodes: a scoping review. Front. Public Health **11** (2023)
41. Ramanathan, A., McDermott, J.: Fall detection with accelerometer data using residual networks adapted to multi-variate time series classification. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2021)
42. Rastogi, S., Singh, J.: A systematic review on machine learning for fall detection system. Comput. Intell. **37**(2), 951–974 (2021)
43. Saleh, M., Abbas, M., Le Jeannes, R.B.: Fallalld: an open dataset of human falls and activities of daily living for classical and deep learning applications. IEEE Sens. J. **21**(2), 1849–1858 (2020)
44. Serramazza, D., Le Nguyen, T., Ifrim, G.: Improving the evaluation and actionability of explanation methods for multivariate time series classification. In: 2024 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2024)
45. Silva, C.A., Casilari, E., García-Bermúdez, R.: Cross-dataset evaluation of wearable fall detection systems using data from real falls and long-term monitoring of daily life. Measurement **235**, 114992 (2024)
46. Son, H., et al.: A machine learning approach for the classification of falls and activities of daily living in agricultural workers. IEEE Access **10**, 77418–77431 (2022)
47. de Sousa, F.A.S.F., Escriba, C., Bravo, E.G.A., Brossa, V., Fourniols, J.Y., Rossi, C.: Wearable pre-impact fall detection system based on 3D accelerometer and subject's height. IEEE Sens. J. **22**(2), 1738–1745 (2021)
48. Stack, E.: Falls are unintentional: studying simulations is a waste of faking time. J. Rehabilitation Assistive Technol. Eng. **4**, 2055668317732945 (2017)
49. Sucerquia, A., López, J.D., Vargas-Bonilla, J.F.: Sisfall: a fall and movement dataset. Sensors **17**(1), 198 (2017)
50. Vaishya, R., Vaish, A.: Falls in older adults are serious. Indian J. Orthopaedics **54**, 69–74 (2020)
51. Wang, G., Li, Q., Wang, L., Zhang, Y., Liu, Z.: Cmfall: a cascade and parallel multi-state fall detection algorithm using waist-mounted tri-axial accelerometer signals. IEEE Trans. Consum. Electron. **66**(3), 261–270 (2020)
52. World Health Organization: Step safely: strategies for preventing and managing falls across the life-course (2021)
53. Wu, X., Zheng, Y., Chu, C.H., Cheng, L., Kim, J.: Applying deep learning technology for automatic fall detection using mobile sensors. Biomed. Signal Process. Control **72**, 103355 (2022)

54. Yi, M.K., Han, K., Hwang, S.O.: Fall detection of the elderly using denoising LSTM-based convolutional variant autoencoder. IEEE Sens. J. (2024)
55. Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. J. Syst. Eng. Electron. **28**(1), 162–169 (2017)

Highly Scalable Time Series Classification for Very Large Datasets

Angus Dempster(✉), Chang Wei Tan, Lynn Miller, Navid Mohammadi Foumani, Daniel F. Schmidt, and Geoffrey I. Webb

Monash University, Melbourne, Australia
angus.dempster@monash.edu

Abstract. Relatively little work in the field of time series classification focuses on learning effectively from very large quantities of data. Large datasets present significant practical challenges in terms of computational cost and memory complexity. We present strategies for extending two recent state-of-the-art methods for time series classification—namely, Hydra and Quant—to very large datasets. This allows for training these methods on large quantities of data with a fixed memory cost, while making effective use of appropriate computational resources. For Hydra, we fit a ridge regression classifier iteratively, using a single pass through the data, integrating the Hydra transform with the process of fitting the ridge regression model, allowing for a fixed memory cost, and allowing almost all computation to be performed on GPU. For Quant, we 'spread' subsets of extremely randomised trees over a given dataset such that each tree is trained using as much data as possible for a given amount of memory while minimising reads from the data, allowing for a simple tradeoff between error and computational cost. This allows for the straightforward application of both methods to very large quantities of data. We demonstrate these approaches with results (including learning curves) on a selection of large datasets with between approximately 85,000 and 47 million training examples.

Keywords: big data · hydra · quant · time series classification

1 Introduction

While some recent work in the field of time series classification focuses on computational efficiency [5,10,11], very little work actually deals with learning effectively from large quantities of data. 'State of the art' in the field of time series classification has come to mean state of the art in terms of accuracy on the datasets in the UCR archive [2,7,23]. Most of these datasets—at least, most of those which are commonly used for evaluation—are small. Median training set size for the 142 canonical univariate datasets is just 217 examples.

This stands in contrast to the datasets commonly used in other domains such as computer vision and natural language processing. It is not coincidental that,

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 80–95, 2025.
https://doi.org/10.1007/978-3-031-77066-1_5

with some exceptions [17,18], deep learning methods have had a relatively muted impact on the field. These are generally low bias methods which require large quantities of training data. The field has also maintained a focus on an 'apples to apples' comparison of compute times based on a fixed amount of compute as a reference point—typically, a single CPU core [23]—potentially overlooking which methods are most efficient in practice, given available hardware (e.g., GPUs).

It may be that the 'bitter lesson'—i.e., that 'the only thing that matters in the long run is the leveraging of computation' [28]—has not yet been learned in time series classification. It may be that methods for time series classification which can exploit GPUs will be more useful in practice for large quantities of data, simply because they are amenable to being implemented in a way that suits available infrastructure per the 'hardware lottery' [15].

Many prominent methods for time series classification are limited by high computational complexity. In other words, these methods are computationally bound to relatively small datasets. However, training on very large quantities of data presents significant practical challenges, even for the most efficient methods, in terms of both memory complexity and computational cost.

Ultimately, memory does not scale with dataset size. It is impractical to require arbitrarily large memory in order to train a given model on increasingly large quantities of data. Further, methods which act on the entire dataset at once are not necessarily more efficient. For example, in practice, 'full' batch gradient descent is significantly less efficient than stochastic gradient descent.

To this end, we present strategies for extending two recent state-of-the-art methods (state of the art on the datasets in the UCR archive) for time series classification—namely, Hydra [10], and Quant [11]—to very large quantities of data. For Hydra, we train a ridge regression classifier iteratively, integrating the transform into the process of fitting ridge regression model. For Quant, we split the data into batches, training a subset of extremely randomised trees on each batch, ensuring that each tree is trained on as much data as possible within a given memory constraint.

The rest of this paper is structured as follows. Section 2 covers relevant related work. Section 3 sets out the strategies for training Hydra and Quant on large datasets. Section 4 presents experimental results, including learning curves, on select large datasets.

2 Background

The preeminence of the datasets in the UCR archive as a basis for benchmarking has meant that the field of time series classification has long been focused on smaller datasets. The field has not been seriously faced with the challenges and tradeoffs inherent in learning from very large quantities of data.

Many of the most prominent methods for time series classification have high computational complexity, requiring significant training time even for relatively small datasets [23]. However, even the most efficient methods face significant practical challenges in training on very large datasets in terms of computational cost and memory complexity.

Additionally, the bias–variance characteristics of methods currently considered state of the art in terms of classification error are likely optimised for smaller datasets, where error is minimised by reducing variance. For larger datasets—where variance decreases as dataset size increases—error is minimised by reducing bias [3].

Most of the datasets in the UCR archive are small. It is therefore reasonable to assume that methods considered state of the art on these datasets are effective at minimising variance. Methods for minimising variance include ensembling (e.g., 'hybrid' methods [22], and methods which use ensembles of decision trees), explicit regularisation (e.g., methods which use ℓ_2 or 'ridge' regularisation), and overparameterisation (e.g., the ROCKET 'family' of methods [8], and other methods which combine a very large feature space with a linear classifier, such as WEASEL 2.0 [27]).

It is as yet unknown which, if any, current state-of-the-art methods are effective when trained on significantly larger quantities of data, or the extent to which these methods represent an appropriate balance of performance versus computational cost on larger datasets compared to, e.g., deep learning methods trained using GPUs.

2.1 HYDRA

HYDRA is a recent method for time series classification focused on computational efficiency. HYDRA combines aspects of ROCKET and dictionary methods [10]. ROCKET transforms input time series using a large number of random convolutional kernels, and then uses the transformed features to train a ridge regression classifier [8]. Dictionary methods involve counting the occurrence of symbolic patterns in time series. HYDRA combines the two approaches, counting the occurrence of random patterns—represented by random convolutional kernels—in the input time series.

Like other members of the ROCKET 'family', HYDRA uses a ridge regression classifier for smaller datasets. For larger datasets, it is intended to be used with logistic regression trained via stochastic gradient descent. In this setting, the time series in each batch are transformed, and the transformed features are used to perform an update step. This allows for a fixed memory cost (proportional to the size of the batch).

However, while reading and transforming the data in batches is memory efficient, computationally it is potentially very inefficient to repeatedly transform the same data over multiple epochs. In contrast to, e.g., a conventional convolutional neural network with learned weights, the convolutional kernels in HYDRA (and other members of the ROCKET 'family') are fixed. One of the main computational advantages of a fixed transform is being able to transform the data only once, i.e., to avoid the need to repeatedly transform the same data. If the transform cost is high (e.g., for long time series with a large number of channels), the time and computational cost spent repeatedly transforming the data over multiple epochs might well be better spent learning the weights of the convolutional

kernels in a conventional convolutional neural network, rather than training a linear classifier on features produced by a fixed transform.

One approach is to store or cache the transformed features, so that each time series need only be transformed once [10]. For small datasets, it makes little difference whether the transformed features are stored or not (as the memory and computational costs are both small). As dataset size grows, caching the transformed features becomes more efficient. However, it becomes infeasible to store the transformed features for arbitrarily large data.

In this context, we present a strategy for training a ridge regression classifier iteratively with a single pass through the data, integrating the Hydra transform with the process of fitting the ridge regression model, obviating the need to store the transformed features.

2.2 Quant

Quant was recently found to be the fastest and most accurate interval method on the datasets in the newly-expanded UCR archive [23]. Interval methods involve computing summary statistics and miscellaneous other features over subseries (intervals) of the input time series [11]. Quant uses a single type of feature (i.e., quantiles), and an 'off the shelf' classifier, namely, extremely randomised trees. Quant has not previously been evaluated on datasets larger than those in the set of 142 datasets used in Middlehurst et al. [23].

There are various established approaches for training ensembles of decision trees on very large quantities of data, including training each tree on a subset of the training data ('pasting') [4], a subset of the features ('random subspaces'), or a subset of both examples and features ('random patches') [19,20].

However, most or all of this work assumes that the ensemble is being trained directly on the underlying data. Here, however, we need to both read and transform the data using the Quant transform, and then train the classifier on the transformed data. As such, we encounter a similar tradeoff between memory complexity and computational cost as for Hydra. It is untenable to cache or store the transformed features for arbitrarily large datasets. It is also computationally undesirable to repeatedly read and transform the same training examples, even if the Quant transform is relatively efficient. At the same time, as the results clearly show, it is desirable to train each tree on as much data as possible: see Sect. 4, below. To this end, we present an approach for training different subsets of trees on batches of data, where each batch is as large as possible subject to memory constraints.

3 Method

3.1 Hydra

The central challenge in extending Hydra (or any other member of the Rocket 'family' of methods) to very large datasets is resolving the tradeoff between

memory complexity and computational cost. We train a ridge regression classifier iteratively, using a single pass through the data, integrating the HYDRA transform into the process of fitting the ridge regression model. This allows for 'the best of both worlds' in the sense that we neither have to repeatedly read and transform the same data (each training example is read and transformed only once), or to store the transformed features (the transformed features are used to update an intermediate quantity used to fit the ridge regression model, and then discarded). Almost all computation can be performed on GPU. The same procedure can be used for any other member of the ROCKET 'family' of methods (or any other fixed transform). However, HYDRA has a significant advantage over other methods in this context, as discussed further below.

Fitting a ridge regression classifier involves fitting a ridge regression model where the classes have been encoded as regression targets, i.e., $\mathbf{Y} \in \{-1, +1\}$. Ridge regression uses a ridge or ℓ_2 penalty (given by λ), and has a closed-form solution [14]:

$$\boldsymbol{\beta}_\lambda = (\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathrm{T}\mathbf{Y}. \tag{1}$$

In the present context, $\mathbf{X}$ is an $n \times p$ feature matrix, representing the transformed features produced by HYDRA, where n is the number of training examples, and p is the number of transformed features.

***p* > *n*.** In practical terms, for transforms such as HYDRA, which produce a relatively small number of features, where $p > n$—i.e., where there are more features than training examples—this implies that the training set is relatively small. Accordingly, where $p > n$, we transform the entire training set, compute $\mathbf{X}\mathbf{X}^\mathrm{T}$, and use the 'shortcut' method (via eigendecomposition of $\mathbf{X}\mathbf{X}^\mathrm{T}$) in order to fit the ridge regression model and estimate LOOCV error for different candidate values of the ridge parameter. Computing $\mathbf{X}\mathbf{X}^\mathrm{T}$ reduces the $n \times p$ feature matrix to an $n \times n$ matrix. This makes the cost of fitting the ridge regression model proportional to n, regardless of the number of features, when $p > n$. The shortcut method has been discussed extensively elsewhere [30].

***n* ≥ *p*.** With respect to (1), both $\mathbf{X}^\mathrm{T}\mathbf{X}$ and $\mathbf{X}^\mathrm{T}\mathbf{Y}$ can be computed iteratively. This follows from the definition of these two quantities:

$$\mathbf{X}^\mathrm{T}\mathbf{X} = \sum_i \mathbf{x}_i^\mathrm{T}\mathbf{x}_i \tag{2}$$

$$\mathbf{X}^\mathrm{T}\mathbf{Y} = \sum_i \mathbf{x}_i^\mathrm{T}\mathbf{y}_i. \tag{3}$$

Accordingly, where $n \geq p$, we read the data in batches, transform each batch using the HYDRA transform, and then update $\mathbf{X}^\mathrm{T}\mathbf{X}$ and $\mathbf{X}^\mathrm{T}\mathbf{Y}$ per (2) and (3). The transformed features can then be discarded. Note also that the ordering of the data is irrelevant. $\mathbf{X}^\mathrm{T}\mathbf{X}$ and $\mathbf{X}^\mathrm{T}\mathbf{Y}$ are of size $p \times p$ and $p \times k$ respectively, where k is the number of classes. This makes the cost of fitting the ridge regression model proportional to p, regardless of n, where $n \geq p$.

While this approach is applicable to any member of the ROCKET 'family' of methods (or, indeed, any fixed transform), HYDRA has a significant advantage in this context. While ROCKET, MINIROCKET [9], and MULTIROCKET [29], for example, all use a fixed number of features (by default, between 10 and 50 thousand), HYDRA computes a number of features proportional to time series length. By default, HYDRA produces 512 features per dilation, where dilation is specified as $d \in 2^{\{0,1,\ldots,\lfloor \log_2 L \rfloor\}}$, where L is time series length. In other words, doubling the length of a time series only increases the number of features by 512. In practice, this means that except in the case of very long time series, HYDRA produces fewer features than the other transforms, and both $\mathbf{X}^\mathrm{T}\mathbf{X}$ and $\mathbf{X}^\mathrm{T}\mathbf{Y}$ ($p \times p$ and $p \times k$ respectively) are meaningfully smaller.

As n grows relative to p, using the shortcut method to compute estimated LOOCV error for different values of λ becomes increasingly less attractive, as it requires computing quantities which are proportional in size to n (fundamentally, it requires computing error per example). Accordingly, for $n \geq p$, we subsample a small validation set (being the smaller of 20% of the training set or 8,192 examples) prior to fitting the model. We choose the value of λ which minimises error on this validation set. (For both the 'shortcut' LOOCV procedure and the separate validation set we perform a grid search over 21 candidate values of λ across 12 orders of magnitude centred on $\sqrt{n}$). In order to compute the inverse of $\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I}$, we compute the eigendecomposition $\mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^\mathrm{T} = \mathbf{X}^\mathrm{T}\mathbf{X}$. The inverse is then given by: $(\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I})^{-1} = \mathbf{V}(\mathbf{\Sigma}^2 + \lambda)^{-1}\mathbf{V}^\mathrm{T}$.

This procedure assumes that $\mathbf{X}$ is centred, and that we fit an intercept term. It is typical to normalise $\mathbf{X}$ by subtracting the mean and dividing by the standard deviation. As we are reading the data in batches, we compute the mean and standard deviation of (the columns of) $\mathbf{X}$, and the mean of $\mathbf{Y}$, as cumulative averages. This is very similar to batch norm [16]. We then compute:

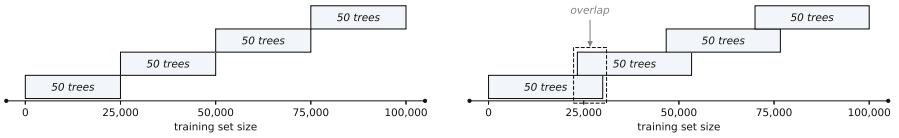
$$\mathbf{M}_\mathbf{X} = \boldsymbol{\mu}_\mathbf{X}^\mathrm{T}\boldsymbol{\mu}_\mathbf{X} \cdot n' \tag{4}$$

$$\boldsymbol{\Phi}_\mathbf{X} = \boldsymbol{\phi}_\mathbf{X}^\mathrm{T}\boldsymbol{\phi}_\mathbf{X} \tag{5}$$

$$\mathbf{M}_{\mathbf{X},\mathbf{y}} = \boldsymbol{\mu}_\mathbf{X}^\mathrm{T}\boldsymbol{\mu}_\mathbf{y} \cdot n' \tag{6}$$

Here, $\boldsymbol{\mu}_\mathbf{X}$ and $\boldsymbol{\phi}_\mathbf{X}$ are row vectors representing the mean and standard deviation of (the columns of) $\mathbf{X}$, and n' is the size of the training set less the size of the validation set. (Accordingly, $\mathbf{M}_\mathbf{X}$ and $\boldsymbol{\Phi}_\mathbf{X}$ are matrices with the same shape as $\mathbf{X}^\mathrm{T}\mathbf{X}$, i.e., $p \times p$). We then normalise $\mathbf{X}^\mathrm{T}\mathbf{X}$ by subtracting $\mathbf{M}_\mathbf{X}$ and dividing by $\boldsymbol{\Phi}_\mathbf{X}$ (elementwise), and normalise $\mathbf{X}^\mathrm{T}\mathbf{Y}$ by subtracting $\mathbf{M}_{\mathbf{X},\mathbf{y}}$ and dividing by $\boldsymbol{\phi}_\mathbf{X}$. The intercept term is $\boldsymbol{\mu}_\mathbf{y}$.

We note that this procedure results in the same model as a 'full' ridge regression fit, i.e., (1): it should produce the same model weights and predictions. However, our approach is both memory and compute efficient, where a 'full' fit is infeasible for very large quantities of data. Importantly, all of these operations—transforming the input time series, forming $\mathbf{X}^\mathrm{T}\mathbf{X}$ and $\mathbf{X}^\mathrm{T}\mathbf{Y}$, computing the eigendecomposition of $\mathbf{X}^\mathrm{T}\mathbf{X}$, fitting the model for different candidate values of λ and estimating error on the validation set—can be performed on GPU.



(a) Training set divisible by batch size. (b) Training set not divisible by batch size.

Fig. 1. Batch size represents the maximum amount of data which can be read, transformed, and used to train trees for a given memory constraint. We divide the total number of trees between the batches and train a subset of trees on each batch. (If training set size is not divisible by batch size, the batches will overlap). We shuffle the underlying training data such that each batch represents a random sample.

Multivariate Time Series. Multivariate time series consist of time series with multiple channels, i.e., for a time series $\mathbf{Z} = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{L-1})$, each $\mathbf{z}_i$ is a vector of values, $\mathbf{z}_i = (z_{i,0}, z_{i,1}, z_{i,C-1})$ for C channels. For multivariate time series data, we use the original multivariate implementation of HYDRA [10]. A different random subset of between 2 and 8 channels is assigned to each group of kernels. In effect, each group of kernels is applied to a univariate time series being the sum of a random subset of channels, with a different subset being used for each group. Following the distributive property of convolution, i.e., $X_0 * W + X_1 * W \equiv (X_0 + X_1) * W$, this is equivalent to applying the same kernel to each channel in the combination. This is conceptually similar to the multivariate implementation of MINIROCKET [9].

3.2 QUANT

For QUANT, we propose reading and transforming the training set in batches, the batches being as large as possible for a given memory constraint, as illustrated in Fig. 1. We divide the total number of trees between the batches and train a subset of trees on each batch. For example, for a total of 200 trees (the default for QUANT), for four batches, we assign and train 50 trees on each batch. (The memory required per time series increases as the number of channels and time series length increase. For short time series with a small number of channels, this can result in reading very large batches, whereas for longer time series with many channels, this could result in smaller batches). This ensures that: (a) each tree is trained on as much data as possible (subject to available memory); and (b) all of the training data is used at least once, while minimising the need to read and transform training examples. For smaller datasets, this defaults to simply training all trees on all of the training data. Where training set size is not divisible by batch size, the batches will overlap. However, the same training example will appear in at most two batches, so will be read and transformed at most twice. (This could be further optimised by only reading and transforming overlapping sections once). We shuffle the data before forming the batches, such that batches smaller than the size of the training set represent approximately uniform random subsets.

This is effectively a simplified version of 'pasting' [4], where we minimise the number of times each training example is sampled, and maximise the amount of data used to train each tree. Louppe and Geurts [19,20] find that training decision trees on subsets of the training examples and/or subsets of features can produce similar classification error to training an ensemble of decision trees (e.g., extremely randomised trees) on the entire dataset. This is somewhat at odds with our results, which suggest that, on large datasets, error is a function of tree size, and that each tree should be trained on as much data as possible.

A closer examination of the findings in Louppe and Geurts [20] suggests some potential caveats. The actual differences in classification accuracy between different methods is small in absolute terms for most of the datasets evaluated. The largest training set used was comprised of only 35,000 training examples (with 784 features), i.e., the equivalent of only approximately 100 MB of training data (assuming float32). Most importantly, for the experiments on the three largest datasets, ensemble size was reduced and tree depth was limited, meaning that there is no real comparison to a model with trees trained to full depth on all the training data.

Pasting was originally proposed in a very different context in terms of the available computing resources. Nevertheless, the results in Breiman [4] in relation to pasting appear to be consistent with our results, i.e., that broadly speaking error is proportional to the quantity of data used to train each tree.

Our results clearly show that model complexity and 0–1 loss on large datasets are bounded by the quantity of data used to train each tree. There is a clear tradeoff between performance (0–1 loss) and computational and memory complexity. In terms of minimising 0–1 loss, it is clear that each tree should be trained on as much data as possible (within memory and computational constraints). This suggests that, ideally, all trees should be trained on all data, i.e., updating each tree using the data in each batch or, in other words, separating the question of memory complexity and batch size. We leave this for future work.

Multivariate Time Series. The original implementation of Quant extends trivially to multivariate data, as the quantiles are already computed over each channel in a multivariate time series. The quantiles computed for each channel can be reshaped into a single set of features per time series (i.e., combining the quantiles from all channels). Indeed this appears to be how multivariate data is handled for the implementation of Quant available in the aeon toolkit [31]. Whether or not this is an optimal strategy for learning from multivariate data, both in terms of absolute performance (e.g., 0–1 loss), or computationally (in terms of the trade off between error and computational complexity), is untested.

4 Experiments

We demonstrate these approaches using five large datasets—*MosquitoSound*, *Pedestrian*, *S2Agri*, *Traffic*, and *WhaleSounds*—with a total of between approximately 105,000 and 59 million examples. In each case, we use 5-fold cross-validation (using predefined folds) such that, for each fold, approximately 80% of

the data is used for training and 20% of the data is used for evaluation. Results presented here in terms of 0–1 loss and training time represent mean 0–1 loss and mean training time over the cross-validation folds.

We present results for two versions of QUANT: (a) with a maximum batch size of 100 MB; and (b) with a maximum batch size of 1 GB. We compare the results for QUANT and HYDRA to results for DrCIF and HInceptionTime on four of the five datasets. We also present results for the datasets in the UEA multivariate time series classification archive [1].

Unless otherwise stated, QUANT is trained using 4 CPU cores, and HYDRA is trained using GPUs, on a cluster with Intel Xeon Gold CPUs and a mixture of NVIDIA V100, A40, and A100 GPUs (almost all jobs used V100 GPUs). Both methods are implemented in Python. Our code is available at: https://github.com/angus924/aaltd2024.

4.1 Datasets

MosquitoSound, taken from the broader UCR archive, consists of 279,566 (univariate) time series, each of length 3,750, representing recordings of wingbeats for six different species of mosquito [12]. The task is to identify the species of mosquito based on the recordings. The dataset has been split into stratified random cross-validation folds.

Pedestrian represents hourly pedestrian counts at various locations in Melbourne, Australia between 2009 and 2022 [6]. The processed dataset consists of 189,621 (univariate) time series, each of length 24. The task is to identify location based on the time series of counts. The dataset has been split into stratified random cross-validation folds.

S2Agri consists of pixel-level Sentinel-2 data at a 10 m resolution [13,26]. The processed dataset contains 59,628,823 multivariate time series, with 10 channels (representing 10 spectral bands), each of length 24. This version of the dataset contains 34 classes representing different land cover types. The dataset has been split into cross-validation folds based on geographic location.

Traffic consists of hourly traffic counts at various locations in the state of NSW, Australia [32]. The processed dataset contains 1,460,968 (univariate) time series, each of length 24. The task is to predict the day of the week based on the time series of counts. The dataset has been split into stratified random cross-validation folds.

WhaleSounds consists of underwater acoustic recordings around Antarctica, manually annotated for seven different types of whale calls [24,25]. The dataset has been processed to extract the annotated whale calls from the original recordings. The processed dataset contains 105,163 (univariate) time series, each of length 2,500, with eight classes representing the seven types of whale call plus a class for unidentified/ambiguous sounds. This version of the dataset has been split into stratified random cross-validation folds.

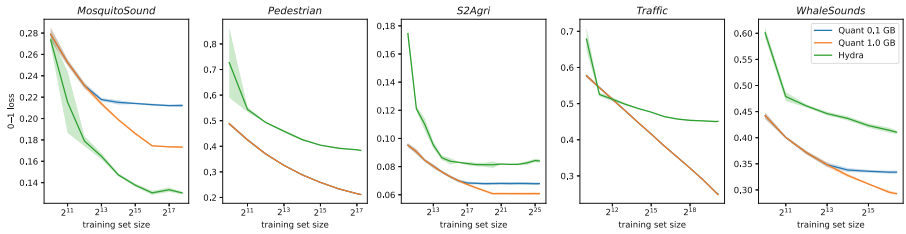


Fig. 2. Learning curves (0–1 loss) for QUANT (for batch sizes of 0.1 GB and 1.0 GB) and HYDRA on *MosquitoSound*, *Pedestrian*, *S2Agri*, *Traffic*, and *WhaleSounds*.

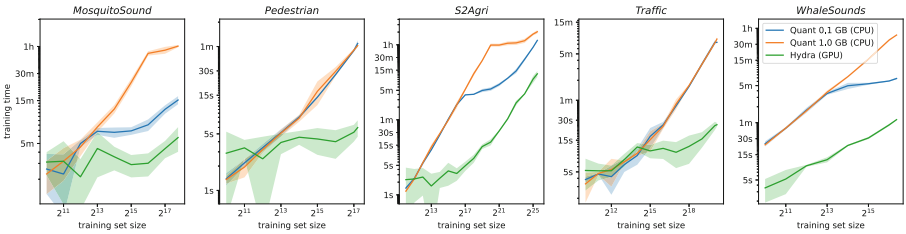


Fig. 3. Training times for QUANT (for batch sizes of 0.1 GB and 1.0 GB) and HYDRA on the same datasets.

4.2 Learning Curves

0–1 Loss. Figure 2 shows learning curves (0–1 loss) for QUANT and HYDRA on *MosquitoSound*, *Pedestrian*, *S2Agri*, *Traffic*, and *WhaleSounds*. Figure 3 shows the corresponding training times for each method. Figure 2 shows that while HYDRA achieves lower 0–1 loss on *MosquitoSound*, QUANT achieves lower 0–1 loss on *Pedestrian*, *S2Agri*, *Traffic*, and *WhaleSounds*. (*MosquitoSound* is comprised of relatively long time series. The default hyperparameter settings for QUANT in terms of the number of intervals and the number of quantiles per interval may be suboptimal in this context). *Pedestrian*, *S2Agri*, and *Traffic* are all similar in that they are comprised of relatively very short time series. We hesitate to draw firm conclusions in relation to the relative performance of the two methods on this sample of datasets.

Figure 2 shows that, for HYDRA, 0–1 loss continues to decrease (even if only modestly) even at the largest training set sizes, with the exception of *S2Agri*. (The small 'uptick' in error at the largest training set sizes for *S2Agri* appears to be due to a shortcoming in the process for determining the regularisation parameter. We leave the resolution of this point for future work). We note that the combination of HYDRA and logistic regression may achieve lower 0–1 loss (at the expense of additional computational and/or memory complexity) as logistic regression is a more 'flexible' model, particularly where $n > p$. While both linear models, ridge regression is limited to models satisfying (1), whereas logistic regression is not. We leave a comparison of the two approaches to future work.

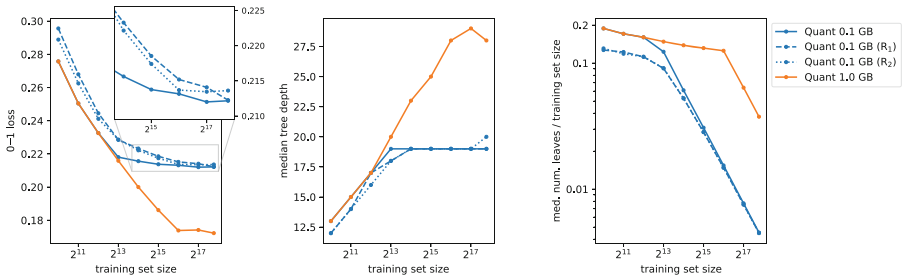


Fig. 4. 0–1 loss (left), median tree depth (centre), and the ratio of the median number of leaves per tree to training set size (right) for QUANT (for maximum batch sizes of 0.1 GB and 1.0 GB) on *MosquitoSound*.

Figure 2 shows that, for QUANT, 0–1 loss continues to decrease, even at the largest training set sizes, for *Pedestrian* and *Traffic*, but appears to largely stop decreasing, or decreases much more slowly after a certain point, for *MosquitoSound*, *S2Agri*, and *WhaleSounds*. This is strongly connected to the quantity of data used to train each tree (see further below).

Training Times. Figure 3 shows the corresponding training times for each method. In one sense, the training times for QUANT and HYDRA are not comparable, as QUANT is trained using CPUs, and HYDRA is trained using GPUs. Ultimately, however, HYDRA is able to make use of GPUs for training, whereas QUANT is not (at least as currently implemented). As such, these results provide an indication of real-world training times for these methods on datasets of this size. At this scale, it is impractical to train different methods in such a way as to allow for a direct, 'apples to apples' comparison of training times, i.e., by training each method on a small fixed number of CPU cores.

Figure 3 shows that, for both HYDRA and QUANT, training time is essentially linear with training set size. Training times for HYDRA are low in absolute terms, and relatively 'flat' on three of the datasets. On this point, training times for HYDRA are affected by the relative size of n vs p (the computational cost of fitting the ridge regression model being proportional to $\min(n, p)$: see Sect. 3), the computational cost for the different validation schemes, and the relative expense of the transform versus the cost of fitting the ridge regression classifier.

For QUANT, training time is dominated by the quantity of data used to train each tree. Once each batch reaches its maximum size (i.e., either 0.1 GB or 1.0 GB), the cost of training the ensemble on more data (but with the same maximum amount of data per tree) declines.

Model Complexity (QUANT). Figure 4 shows 0–1 loss (left), median tree depth (centre), and the ratio of the median number of leaves per tree to training set size (right) for QUANT, for maximum batch sizes of 0.1 GB and 1.0 GB, on

Table 1. 0–1 loss for QUANT, HYDRA, HInceptionTime, and DrCIF.

	$\text{QUANT}_{0.1}$	$\text{QUANT}_{1.0}$	HYDRA	HInception	DrCIF (4h)	DrCIF (8h)
MosquitoSound	0.2119	0.1731	0.1304	0.1743	0.2998	0.2394
Pedestrian	0.2115	0.2115	0.3856	0.3319	0.2228	0.2112
S2Agri	0.0675	0.0608	0.0845	–	–	–
Traffic	0.2508	0.2481	0.4508	0.3345	0.3667	0.3310
WhaleSounds	0.3339	0.2928	0.4092	0.4867	0.3522	0.3207

Table 2. Training times (excluding *S2Agri*).

$\text{QUANT}_{0.1}$	$\text{QUANT}_{1.0}$	HYDRA	HInception	DrCIF (4h)	DrCIF (8h)
32 m 17 s	1h 57 m	8 m 12 s	6 d	18 h 25 m	1 d 9 h

MosquitoSound (for a single fold). Figure 4 also shows results for two different methods for sampling batches with replacement ('R_1' and 'R_2').

Figure 4 shows very clearly that 0–1 loss is closely tied to model size, which is a function of the quantity of data used to train each tree. More data per tree allows for training deeper trees which, in turn, results in lower 0–1 loss. In other words, the quantity of data used to train each tree represents a kind of floor on 0–1 loss. While 0–1 loss continues to decrease modestly after reaching maximum batch size, there is a clear advantage to simply using more data.

The results are essentially the same when sampling batches with replacement (equivalent to bagging when training set size is smaller than maximum batch size and, in the limit, similar to sampling without replacement, to the extent that the probability of sampling the same object twice from a large set becomes increasingly small). Here we show two variants of sampling with replacement: using the same size and number of batches as the default, but sampling each batch with replacement ('R_1'), and always sampling at least 50 batches (of up to maximum batch size, i.e., always training 4 trees per batch) and forming each batch with replacement ('R_2').

Figure 4 (right) shows that model size grows approximately linearly (training set size is approximately 5× to 10× the median number of leaves), until maximum batch size is reached, at which point model size stays essentially the same, while the quantity of data continues to increase. While the trees become relatively deep (with a median depth of approximately 28 by 2^{17} training examples), the actual model size (total number of nodes) is significantly smaller than a 'full' tree of the given depth, i.e., at 2^{17} training examples, for a median depth of 28, the median node count is just 16,740, i.e., closer to $\sqrt{2^{28}}$.

It seems clear that it would be advantageous to be able to continue training each tree on all data, subject to memory and computational constraints. We leave this for future work.

4.3 Comparisons with Other Methods

We present a comparison against preliminary results for DrCIF and HInception-Time on four of the five datasets. (Training DrCIF and HInceptionTime on the

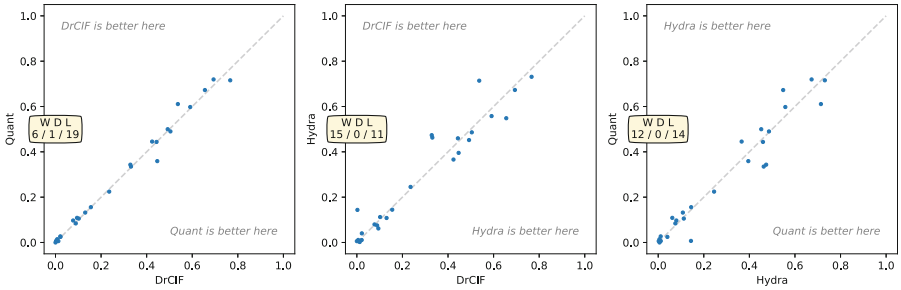


Fig. 5. Pairwise 0–1 loss for QUANT vs DrCIF (left), HYDRA vs DrCIF (centre), and QUANT vs HYDRA (right) on 26 datasets from the UEA multivariate time series classification archive.

S2Agri dataset is beyond the scope of this paper). DrCIF is an interval method which achieves broadly similar results to QUANT on the datasets in the UCR archive [21–23]. HInceptionTime is an ensemble of convolutional neural network models and represents the most accurate deep learning model on the datasets in the UCR archive [17,23].

Table 1 shows 0–1 loss for QUANT and HYDRA versus DrCIF and HInceptionTime for the full training set for four of the five datasets. With one exception, the results presented here represent mean results over five cross-validation folds. (Due to time and computational constraints, the results for HInceptionTime on *MosquitoSound* represent mean results over two folds).

Table 1 shows that HInceptionTime achieves lower 0–1 loss than HYDRA on two of the four datasets, but that QUANT achieves lower 0–1 loss than HInceptionTime on all four datasets. DrCIF achieves lower 0–1 loss than HYDRA on three of the four datasets, and lower 0–1 loss than QUANT on one dataset.

Table 2 shows the total training time for each method (excluding *S2Agri*), averaged over the relevant folds. While these times are not directly comparable, it is clear that the outlier is HInceptionTime. HInceptionTime was trained using GPUs. DrCIF was trained using a single core per run using two different contract times: 4h and 8h. (Training runs for DrCIF did not reliably complete when using multiple threads/cores). Error is clearly decreasing as DrCIF is allowed more time to train. Given the similarity between the results for DrCIF and QUANT on the datasets in the UCR archive [23], and the UEA multivariate time series archive (see below), we expect that, with sufficient training time, 0–1 loss for DrCIF would at least match that for QUANT on these datasets.

4.4 Multivariate Datasets

Results on the datasets in the UEA multivariate time series archive have not previously been presented for HYDRA or QUANT. Accordingly, we take the opportunity to present such results here, as they serve as a useful reference point for work involving multivariate time series data. Figure 5 shows pairwise 0–1 loss for

Quant vs DrCIF (left), Hydra vs DrCIF (centre), and Quant vs Hydra (right). (Results for DrCIF are taken from Middlehurst et al. [22]). In relation to Quant, given the relatively small size of these datasets, increasing the batch size beyond approximately 100 MB makes essentially no difference to the results (as almost all the datasets are smaller than 100 MB). As such, we show results for Quant with a maximum batch size of 100 MB (this is essentially the default configuration). Figure 5 shows that, consistent with recent results on the expanded set of univariate datasets from the UCR archive [23], 0–1 loss for Quant and DrCIF is very similar for most datasets. DrCIF achieves lower 0–1 loss than Quant on 19 datasets, but the differences are mostly small. In contrast, 0–1 loss for Hydra and DrCIF (and, by implication, Quant) is less correlated. Hydra achieves lower 0–1 loss than DrCIF on 15 datasets, although DrCIF achieves noticeably lower 0–1 loss on four of the datasets. Total training time (averaged over 30 folds) is 16 min 6 s for Quant (CPU), and 1 min 12 s for Hydra (GPU).

5 Conclusion

The field of time series classification has long been focused on smaller datasets. Even in relation to more efficient methods, very little attention has been paid to learning effectively from very large quantities of data. Learning from large datasets requires making effective use of available computational resources, and operating in a context where dataset size might be considerably larger than available memory.

We present strategies for applying two state-of-the-art methods—Hydra and Quant—to large quantities of data. It is clear that in practical terms, Hydra—trained using GPUs—is considerably faster than Quant (by a large constant factor), although the difference could be reduced by using more CPU cores for Quant where possible. However, Quant achieves lower 0–1 loss than Hydra on four of the five large datasets included in this study.

There are several important limitations to the strategies presented here. The efficient iterative method of fitting the ridge regression classifier is limited to linear (ridge) regression models. The additional flexibility of gradient descent (e.g., for training nonlinear models) is likely to allow for achieving lower 0–1 loss on larger datasets, albeit with additional computational cost and/or memory complexity. For Quant, the results show that it is clearly desirable to train each tree with as much data as possible (subject to compute and memory constraints). The quantity of data used to train each tree forms a ceiling for model complexity and therefore the ability to learn from additional data. Nevertheless, we hope that the approaches set out here can help to form an efficient baseline for performance versus computational cost on larger datasets.

Acknowledgments. This work was supported by the Australian Research Council under award DP240100048.

References

1. Bagnall, A., et al.: The UEA multivariate time series classification archive. arXiv:1811.00075 (2018)
2. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Disc. **31**(3), 606–660 (2017)
3. Brain, D., Webb, G.I.: The need for low bias algorithms in classification learning from large data sets. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) Principles of Data Mining and Knowledge Discovery, pp. 62–73. Springer, Berlin (2002)
4. Breiman, L.: Pasting small votes for classification in large databases and on-line. Mach. Learn. **36**(1), 85–103 (1999)
5. Cabello, N., Naghizade, E., Qi, J., Kulik, L.: Fast, accurate and explainable time series classification through randomization. Data Min. Knowl. Discov. (2023)
6. City of Melbourne: Pedestrian counting system (2022). https://data.melbourne.vic.gov.au/explore/dataset/pedestrian-counting-system-monthly-counts-per-hour/information/. CC BY 4.0
7. Dau, H.A., et al.: The UCR time series archive. IEEE/CAA J. Autom. Sinica **6**(6), 1293–1305 (2019)
8. Dempster, A., Petitjean, F., Webb, G.I.: ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Disc. **34**(5), 1454–1495 (2020)
9. Dempster, A., Schmidt, D.F., Webb, G.I.: MiniRocket: a very fast (almost) deterministic transform for time series classification. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 248–257. ACM, New York (2021)
10. Dempster, A., Schmidt, D.F., Webb, G.I.: Hydra: competing convolutional kernels for fast and accurate time series classification. Data Min. Knowl. Discov. (2023)
11. Dempster, A., Schmidt, D.F., Webb, G.I.: Quant: a minimalist interval method for time series classification. Data Min. Knowl. Discov. (2024)
12. Fanioudakis, E., Geismar, M., Potamitis, I.: Mosquito wingbeat analysis and classification using deep learning. In: 26th European Signal Processing Conference, pp. 2410–2414 (2018)
13. Garnot, V.S.F., Landrieu, L., Giordano, S., Chehata, N.: Satellite image time series classification with pixel-set encoders and temporal self-attention (2020)
14. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2009)
15. Hooker, S.: The hardware lottery. Commun. ACM **64**(12), 58–65 (2021)
16. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of Machine Learning Research, pp. 448–456 (2015)
17. Ismail-Fawaz, A., Devanne, M., Weber, J., Forestier, G.: Deep learning for time series classification using new hand-crafted convolution filters. In: IEEE International Conference on Big Data, pp. 972–981 (2022)
18. Ismail Fawaz, H., et al.: InceptionTime: finding AlexNet for time series classification. Data Min. Knowl. Disc. **34**(6), 1936–1962 (2020). https://doi.org/10.1007/s10618-020-00710-y
19. Louppe, G.: Understanding random forests: from theory to practice. Ph.D. thesis, University of Liège (2014). arXiv:2305.11921

20. Louppe, G., Geurts, P.: Ensembles on random patches. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) Machine Learning and Knowledge Discovery in Databases, pp. 346–361. Springer, Berlin (2012)
21. Middlehurst, M., Large, J., Bagnall, A.: The canonical interval forest (CIF) classifier for time series classification. In: IEEE International Conference on Big Data, pp. 188–195 (2020)
22. Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., Bagnall, A.: HIVE-COTE 2.0: a new meta ensemble for time series classification. Mach. Learn. **110**, 3211–3243 (2021)
23. Middlehurst, M., Schäfer, P., Bagnall, A.: Bake off redux: a review and experimental evaluation of recent time series classification algorithms. Data Min. Knowl. Discov. (2024)
24. Miller, B.S., et al.: An open access dataset for developing automated detectors of Antarctic baleen whale sounds and performance evaluation of two commonly used detectors. Sci. Rep. **11** (2021)
25. Miller, B.S., Stafford, K.M., Van Opzeeland, I., et al.: Whale sounds (2020). https://data.aad.gov.au/metadata/AcousticTrends_BlueFinLibrary. CC BY 4.0
26. Sainte Fare Garnot, V., Landrieu, L.: S2Agri pixel set (2022). https://zenodo.org/records/5815488. CC BY 4.0
27. Schäfer, P., Leser, U.: WEASEL 2.0: a random dilated dictionary transform for fast, accurate and memory constrained time series classification. Mach. Learn. **112**(12), 4763–4788 (2023)
28. Sutton, R.: The bitter lesson (2019). http://www.incompleteideas.net/IncIdeas/BitterLesson.html
29. Tan, C.W., Dempster, A., Bergmeir, C., Webb, G.I.: MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. Data Min. Knowl. Disc. **36**(5), 1623–1646 (2022)
30. Tew, S., Boley, M., Schmidt, D.F.: Bayes beats cross validation: efficient and accurate ridge regression via expectation maximization. In: 37th Conference on Neural Information Processing Systems (2023)
31. The aeon Developers: aeon (2024). https://github.com/aeon-toolkit/aeon
32. Transport for NSW: NSW road traffic volume counts hourly (2023). https://opendata.dev.transport.nsw.gov.au/dataset/nsw-roads-traffic-volume-counts-api/resource/bca06c7e-30be-4a90-bc8b-c67428c0823a. CC BY 4.0

Classification of Raw MEG/EEG Data with Detach-Rocket Ensemble: An Improved ROCKET Algorithm for Multivariate Time Series Analysis

Adrià Solana[1,2(✉)], Erik Fransén[1,3,4], and Gonzalo Uribarri[1,3,4]

[1] Section of Computational Brain Science, EECS, KTH Royal Institute of Technology, Stockholm, Sweden
{adriasic,erikf,uribarri}@kth.se
[2] Polytechnic University of Catalonia, Barcelona, Spain
adria.solana@estudiantat.upc.edu
[3] DigitalFutures, KTH Royal Institute of Technology, Stockholm, Sweden
[4] Science for Life Laboratory, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract. Multivariate Time Series Classification (MTSC) is a ubiquitous problem in science and engineering, particularly in neuroscience, where most data acquisition modalities involve the simultaneous time-dependent recording of brain activity in multiple brain regions. In recent years, Random Convolutional Kernel models such as ROCKET and MiniRocket have emerged as highly effective time series classification algorithms, capable of achieving state-of-the-art accuracy results with low computational load. Despite their success, these types of models face two major challenges when employed in neuroscience: 1) they struggle to deal with high-dimensional data such as EEG and MEG, and 2) they are difficult to interpret. In this work, we present a novel ROCKET-based algorithm, named Detach-Rocket Ensemble, that is specifically designed to address these two problems in MTSC. Our algorithm leverages pruning to provide an integrated estimation of channel importance, and ensembles to achieve better accuracy and provide a label probability. Using a synthetic multivariate time series classification dataset in which we control the amount of information carried by each of the channels, we first show that our algorithm is able to correctly recover the channel importance for classification. Then, using two real-world datasets, a MEG dataset and an EEG dataset, we show that Detach-Rocket Ensemble is able to provide both interpretable channel relevance and competitive classification accuracy, even when applied directly to the raw brain data, without the need for feature engineering.

Keywords: Multivariate Time Series Classification · EEG · MEG · ROCKET Algorithm

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 96–114, 2025.
https://doi.org/10.1007/978-3-031-77066-1_6

1 Introduction

Multivariate Time Series Classification (MTSC) is a topic of increasing interest as the amount of acquired data and the number of possible applications in science and engineering grow over time. MTSC presents a challenging problem because a Multivariate Time Series (MTS) can exhibit complex patterns that not only span over time within a single channel, but also over time between multiple channels.

The standard approach in the Time Series Classification (TSC) literature is to develop complex architectures aimed at achieving high accuracy on established benchmarks [3,17,21], with the two most notable being the UCR (University of California, Riverside) archive for Univariate Time Series (UTS) and the UEA (University of East Anglia) archive for Multivariate Time Series (MTS) [2,5]. Examples of state-of-the-art models include TS-CHIEF [24], InceptionTime [14], HIVE-COTE v2.0 [16] and Hydra [8]. While these models are effective, their high accuracy is typically achieved at the expense of scalability, interpretability and computational feasibility.

Motivated by the lack of scalable models in the state-of-the-art, the RandOm Convolutional KErnel Transform (ROCKET) algorithms emerged as a lightweight alternative to TSC. This novel class of algorithms offers simple architectures capable of achieving accuracies comparable to much more complex models. There are three main variants of the ROCKET algorithm, namely ROCKET, MiniRocket and MultiRocket [6,7,25]. Despite some differences, all of them share the same core idea. ROCKET works by generating a large number of random convolutional kernels, which are subsequently aggregated into features and used to train a ridge classifier [6]. Rather than relying on a learning process that fits the kernel parameters to the data, ROCKET depends on its extensive number of random features, with the expectation that some of them will carry information relevant to the classification task. Moreover, recent study demonstrates that it is possible to prune features of the model that are not relevant for classification through a process called Sequential Feature Detachment (SFD) [26]. The authors showed that the resulting pruned model, called Detach-Rocket, achieves better accuracy while requiring less than 10% of the original number of features.

The ROCKET algorithm has proven successful in the TSC paradigm, achieving good performance on the UCR and UEA datasets, as well as other real-world applications [27]. The linear nature of its classifier helps to prevent overfitting on datasets with a limited number of instances and enables the model to handle large datasets efficiently due to its lightweight training scheme. However, there are two areas where the ROCKET model shows limitations:

- **Scalability with number of channels.** To deal with MTS, the original ROCKET algorithm combines all channels for each convolution. This means that each kernel has $L \times C + 1$ coefficients, where L is the length of the kernel and C is the number of channels in the input time series. This simple multivariate strategy faces two problems when C is large. First, the probability that a randomly sampled kernel will encode meaningful multichannel information decays rapidly as the number of interacting channels increases. This means that it will be increasingly difficult for the model to handle higher order

interactions between channels as the number of channels grows. Second, even when the kernel encodes meaningful information for a subset of the channels, the contribution of the remaining channels to this convolution will become an obstacle to classification. These drawbacks have been partially addressed in MiniRocket and MultiRocket by limiting the number of channels mixed by each convolutional kernel, thus limiting the maximum possible order of channel interactions. A downside of this strategy is that each kernel now includes only a random subset of channels, which for large C introduces a lot of variability between different realizations of the model and requires a very large number of kernels to properly sample all possible channel combinations.

– **Interpretability.** Despite their simple architecture, ROCKET models are difficult to interpret. Some of the limitations in this regard are: a) the generated feature space is very high dimensional, b) there is no built-in way to have a label probability, and c) in the case of MTSC, there is no easy way to compute the channel importance (i.e., the relevance of each channel to the classification task).

These limitations are particularly notable when the models are applied to neural data. Noninvasive brain activity is typically measured using inherently multivariate modalities, such as electroencephalograms (EEG) [4], magnetoencephalograms (MEG) [12], and functional magnetic resonance imaging (fMRI) scans [11]. In most cases, these multivariate time series have a large number of channels representing either different sensors or regions of interest (ROIs). In addition, in neuroscience, it is usually important to identify which brain regions are most relevant for discriminating between the different classes, defined typically as experimental conditions, groups of participants, etc. Furthermore, when the model is used for diagnosis, having a reliable measure of label probability is essential for evaluating the robustness of the classifier and for setting a threshold that balances the ratio of false positives to false negatives. Thus, even though ROCKET models have shown some potential for neuroscience tasks [20,22], there are fundamental limitations of the methodology that need to be addressed for this type of data.

In this work, we introduce a novel ROCKET-based model, named Detach-Rocket Ensemble, specifically designed to address multivariate time series classification problems. Our methodology involves creating an ensemble of Detach-MiniRockets, which are pruned MiniRocket models. The ensemble mitigates model variability [9] and enables the exploration of a much larger pool of kernels than a single MiniRocket, increasing the probability of finding meaningful interactions between channels. Due to the small size of the pruned Detach-MiniRocket models, the resulting ensemble is typically smaller than a single MiniRocket. Another advantage of our model is that, as an ensemble, it provides a built-in measure of label probability. In addition, by leveraging the pruning process of Detach-Rocket, we have also developed a straightforward method for estimating channel relevance for the classification task.

To evaluate the channel relevance estimation of our model, we first designed a synthetic MTSC dataset for which we are able to control the amount of information about the classification label carried by each channel of a multivariate time

series. We show that our methodology is able to identify the relevant channels and their relative importance for the classification task.

We then demonstrate the potential of our model on two real-world neuroscience applications. The first consists of a face recognition task using 306-channel MEG data. We show that our model is able to achieve better accuracy on this MTSC dataset than previous ROCKET-based models, and is also able to identify the relevant brain areas for classification, which are consistent with those found in the neuroscience literature for the same task. The second is an Alzheimer's disease classification task using resting-state EEG. We show that our model significantly outperforms state-of-the-art models designed for raw EEG classification, achieving an accuracy comparable to that obtained by applying feature engineering specifically designed for this task. For this classification task, we compute the ROC curve of our model and select the optimal threshold, demonstrating the benefits of incorporating a label probability.

The rest of the paper is organized as follows. In Sect. 2, Background, we review previous ROCKET algorithms relevant to this work, with a particular focus on Detach-Rocket. In Sect. 3, Methods, we introduce the Detach-Rocket Ensemble model and the methodology for estimating channel relevance. In Sect. 4, Synthetic Dataset, we present the results of the synthetic dataset experiment. In Sect. 5, Real Datasets, we demonstrate the application of the Detach-Rocket Ensemble to EEG and MEG datasets. Section 6 contains the discussion, and Sect. 7 presents the conclusions of this study.

2 Background

2.1 Random Convolutional Kernel Transform (ROCKET)

The standard ROCKET model use 10,000 random convolutional kernels to compensate for the lack of a training process typically used on architectures such as Convolutional Neural Networks (CNNs) [1]. These kernels convolute the normalized input time series to extract informative features. To achieve this, the kernel parameters such as the length, weights and dilation are drawn from random distributions.

The convolution of each kernel over a time series results in a feature map, equivalent to a univariate time series that encodes the extracted characteristics. Each of these feature maps are pooled into two scalars through two operators: global max pooling, that selects the maximum value, and the Percentage of Positive Values (PPV), that computes the fraction of values larger than zero. The 20,000 resulting features fit a classifier, usually based on a ridge regression model, that yields a label prediction for the given time series.

2.2 MiniRocket

MiniRocket [7] was developed as a faster and lighter version of the original ROCKET algorithm. This enhanced architecture is more deterministic when

generating the kernels and also eliminates the max pooling operator, thus relying solely on the 10,000 PPVs. These modifications simplify the model and yield slightly superior performance compared to ROCKET in the UCR archive [7].

The MiniRocket parameter selection is fully deterministic, except for the bias calculation and the channel selection. The set of biases to be used in prediction is computed after an initial convolution of a subset of input samples, where each bias is obtained as a quantile from the resulting feature maps. Hence, fully deterministic biases can be obtained if the full training set is used.

For MTS, MiniRocket chooses for each kernel combinations from 1 up to 9 channels with an exponentially decaying distribution, i.e. convolutions are more likely to use less number of channels. The channels that are used for each kernel are chosen with equal probability. In the context of our work, this channel sampling is crucial for studying the relation between the best performing features and the channels that generated them, which is key for the interpretability of MTS.

2.3 Arsenal

The state of the art of TSC is currently dominated by HIVE-COTE v2 [16], a meta ensemble that aggregates different feature extraction and classification modules into a single prediction, achieving cutting-edge accuracies [17]. One of these modules is Arsenal, an ensemble of 25 ROCKET models with 2,000 kernels each. Although Arsenal serves as a component within the full model, it can function as a standalone classifier. In this case, the algorithm works by cross-validating every ROCKET model with the training dataset and producing estimates via soft-voting the predictions, weighted by the scores of each model during cross-validation. The ensemble methodology aims to improve the predictive power of the classifier as well as giving a probability estimate.

2.4 Detach-Rocket

ROCKET provides a large number of features with the objective of training a simple linear classifier for a relatively low cost. However, there is no straightforward method to decide which features are truly relevant, and which ones solely add undesired complexity or even hinder the classification. A previous study presented Sequential Feature Detachment (SFD) [26], a novel algorithm that intends to prune features by keeping the most essential ones (Fig. 1). It does so by iteratively fitting a ridge classifier, pruning a percentage of the features associated with the coefficients of lesser magnitudes, and fitting it again until the desired percentage of features is achieved.

In this study, the efficacy of this methodology was evaluated on all binary classification datasets from the UCR archive. The results demonstrated that a full ROCKET model could be pruned to 2% of its original features while maintaining overall classification accuracy [26]. These findings indicate that SFD can effectively reduce the complexity of a model while enhancing its generalization capabilities.

Furthermore, the authors proposed an end-to-end model named Detach-Rocket that automatically determines how many features should be kept in a pruned model. This method employs a trade-off hyperparameter c, which weights the retention of features against the accuracy of the pruned model. As c approaches 0, greater significance is given to accuracy, while larger values prioritize maintaining a smaller size. The default value suggested for c is 0.1, which gives more importance to accuracy and prunes a model down to roughly 1% in mean according to the tests on the UCR archive.

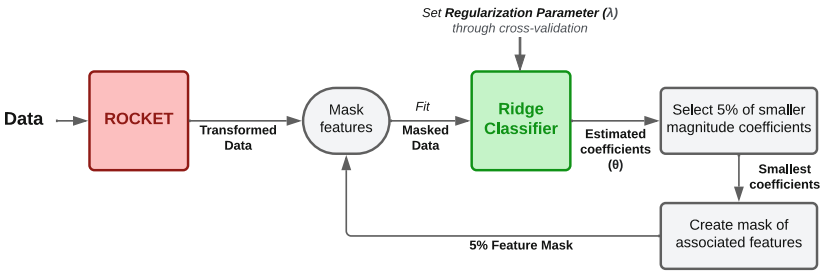


Fig. 1. Sequential Feature Detachment (SFD) diagram.

3 Methods

3.1 Detach-Rocket Ensemble Model

Motivation. As discussed in the Introduction section, when the number of channels C of an MTS is high, ROCKET-based models require a larger sample of convolutional kernels to properly sample the space of possible patterns. One way to do this is to simply increase the number of kernels in a single model. While this strategy may be effective, it is computationally inefficient. First, the forward pass or inference in the model will become slower as the number of convolutions required increases. Second, for a very large number of kernels, it will be impossible to fit the entire feature matrix of size (num. instances, num. features) into RAM, making training either much slower or infeasible.

As a solution to this problem, we introduce Detach-Rocket Ensemble. Instead of training a single model with a large number of kernels, we propose to train several standard models and then aggregate them into an ensemble. This makes the training process both manageable in terms of feature matrix size and trivially parallelizable, while still enabling the exploration of a large pool of kernels.

A key aspect of Detach-Rocket Ensemble is that it is composed of pruned Detach models. These pruned models have been shown to achieve the same classification accuracy as the full model while retaining only 2% of the original features [26]. Consequently, although a large number of kernels are explored during training and pruning, the total number of kernels in the resulting ensemble model is comparable to that in a single standard ROCKET, thereby requiring approximately the same convolutions for inference.

In addition to potentially improving the classification accuracy, an ensemble model provides a reliable class probability, which is useful to explore different thresholds that may better suit the classification task. Furthermore, in our case, the ensemble also helps to achieve a better estimation of channel relevance, as discussed in the next section.

Model Description. A Detach-Rocket Ensemble model is composed of N Detach-MiniRocket models, which are first trained independently. The training process for each model follows the procedure described in Sect. 2.4: Each MiniRocket is pruned with SFD, using a subset of the training set to determine the optimal pruning size. Each of the resulting Detach-MiniRockets is then assigned a weight in the ensemble according to its performance on the training set. A schematic of this training process for a Detach-Rocket Ensemble model with $N = 2$ number of models is presented in Fig. 2.

To perform a prediction on a given instance, the model weights the label prediction made by each Detach-MiniRocket with the corresponding model weight and generates a normalized label probability. This probability can be subsequently thresholded to obtain the final model decision on that instance. The prediction process for a Detach-Rocket Ensemble model with $N = 2$ is also illustrated in Fig. 2.

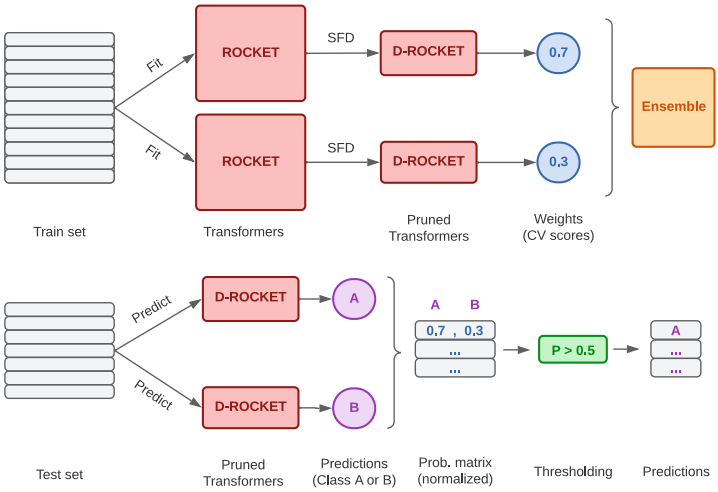


Fig. 2. Methodology of the Detach-Rocket Ensemble, fit (top) and predict (bottom) with N = 2 models and two classes (A and B). D-Rocket is the abbreviation of a single Detach-MiniRocket classifier.

In the present work, we chose Detach-MiniRocket as the base classifier for our ensemble because it is the fastest ROCKET variant. However, the exact same methodology can be applied using Detach-MultiRocket, which, given its

stronger solo performance, may yield even better results for some datasets. It is important to note that, for large C, the variability across the learners arises mainly from the random subset of channels selected for each of the kernels. Although it is possible, we do not recommend using the standard ROCKET as the base classifier due to the reasons discussed in the introduction section and its slower training time.

3.2 Channel Relevance Estimation

To estimate the feature relevance in a Detach-Rocket Ensemble model, we first compute it on each of its constitutive base classifiers. The methodology for estimating relative channel relevance on a single Detach-MiniRocket is illustrated in Fig. 3 and consists of the following steps:

1. SFD selects the most relevant kernels in the MiniRocket model.
2. The model retrieves the channels that each selected kernel has used.
3. Each channel is given an importance proportional to the weight (θ_i) of its kernel divided by the number of channels in the same kernel.
4. The pondered channels are summed and normalized into a relative relevance histogram.

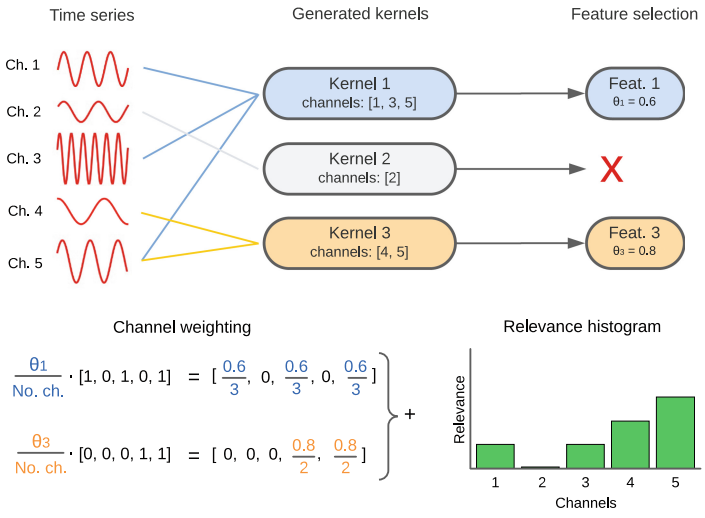


Fig. 3. Methodology employed to estimate the relevance of each channel using Detach-Rocket. In this example, three kernels are applied over a time series comprising five channels. Subsequently, SFD selects the first and third features, thereby selecting the first and third kernels. The channels combined by these kernels are then weighted and accounted for in the computation of the channel relevance histogram.

The pruning process is the essential part of this methodology, as it selects the relevant kernels and completely removes the irrelevant ones from the channel

importance calculation, which is the majority of them. The subsequent weighting, based on the coefficient of the ridge classifier and the number of channels involved, is a way to achieve a greater sensitivity in the identification of the relevant channels. This methodology has been developed and optimized through a series of tests on a fully controlled synthetic dataset.

Once the relative channel relevance has been computed for all base classifiers, the resulting channel relevance for the ensemble model is determined by taking the median of the relevancies obtained for each channel in the base models and normalizing along the channels. In addition to potentially improving classification accuracy, the ensemble mitigates the variability in the base models' channel relevance estimates, resulting in a more reliable estimate.

Note that the proposed channel relevance estimation is built-in, meaning that it does not require additional computation for strategies such as predicting instances with masked channels or retraining the algorithm on a subset of channels.

4 Synthetic Dataset

4.1 Synthetic Dataset Design

In this first experiment, we created a synthetic time series classification dataset to test our proposed channel relevance estimation procedure. The goal was to have a controlled environment where the contribution of each channel to the classification task is known, so that we can compare the estimated channel relevance to a ground truth.

To this end, we constructed a four-dimensional binary time series dataset. This synthetic dataset is designed so that one parameter, theta (θ), controls how the information for classification is distributed along two of the four channels. The remaining two channels are noninformative channels used as control. Specifically, the description of the channel content is as follows: Channels 1 and 2 contain two different sinusoidal waveforms whose amplitude values are relevant to the classification task, Channel 3 contains a sinusoidal waveform whose amplitude is not relevant to the classification task, and Channel 4 contains no waveform. All channels also contain additive white noise. Figure 4(C) presents a representation of two samples of this dataset, one from each of the classes.

For a given value of theta, the dataset is created by sampling the values of the sinusoidal amplitudes A1 and A2 in channels 1 and 2 respectively from a pair of Gaussian distributions, one belonging to each of the two classes. Theta controls the positioning of these distributions in the parameter plane, the plane of all possible sinusoidal amplitudes as depicted in Fig. 4(A). See also Fig. 4(B) for a particular sampling instance. The theta angle controls the relative position of the Gaussian distributions with respect to the amplitude plane, thus determining the amount of information about the classification task carried by each of the amplitudes. If theta is 45°, then the class label information is equally distributed between channels 1 and 2. If theta is zero, then all the information about the

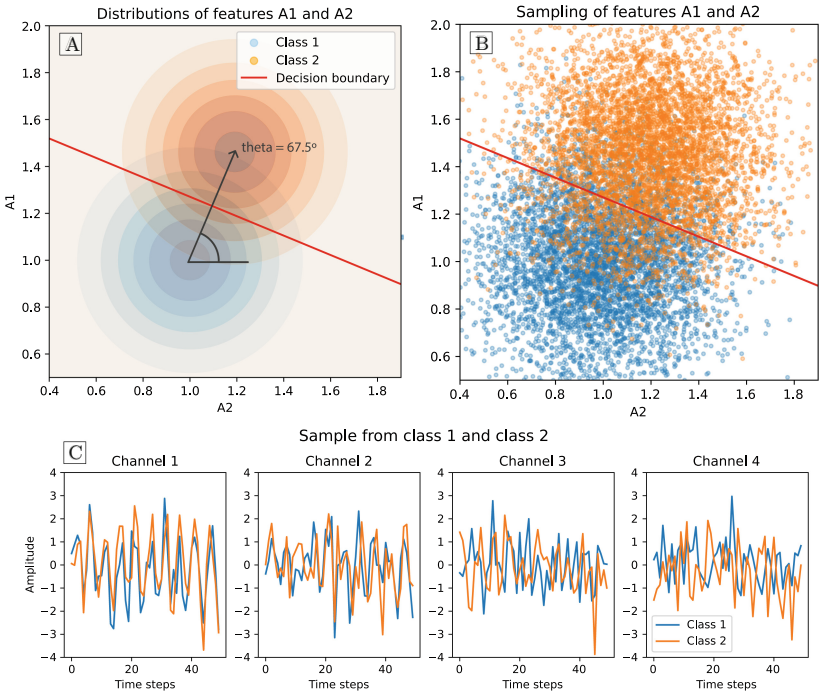


Fig. 4. Example plots depicting the synthetic dataset. Plot A shows the Gaussian distributions from which the sinusoidal amplitudes of channels 1 and 2 (A1 and A2) are sampled, for both classes, in addition to the analytical ideal decision boundary. It also shows the effect of theta, that shifts the center of the distribution of class 2 around class 1. Plot B shows a sampling instance of these distributions. Plot C presents one sample of each class, both unraveled along their 4 channels, after the amplitudes have been sampled and white noise has been added.

class is carried by channel 2, while if theta is 90°, all the information is carried by channel 1.

To estimate the difficulty of this classification task, we calculated the optimal decision boundary, depicted in red in Fig. 4A and B. This determines the maximum achievable accuracy when classifying in the parameter space, constituted by the first and second channels amplitudes. The analytical results demonstrated that, for all values of theta, this accuracy was 84.13%. The complexity of the actual classification task is much higher, since these amplitude parameters are then transformed into waveforms, two interfering channels are included, and high-power white noise is added to the time series. Figure 4C shows two instances of the synthetic dataset that illustrate the non-triviality of the classification task.

4.2 Synthetic Dataset Results

We explored the relevance of the channels for several values of theta ranging from 0° to 90°. For each one of the resulting datasets, we used a Detach-Rocket Ensemble of 25 models with 10,000 kernels each to estimate the channel relevance. The dataset design is such that the first channel -with a sinusoid of amplitude A1- is irrelevant when $\theta = 0°$, and that it increases with theta until it is the only important channel at $\theta = 90°$. The opposite happens with the second channel, with a sinusoid of amplitude A2. While the relevance values of the channels are not strictly defined at intermediate thetas, they should be equal at $\theta = 45°$, and they should monotonically increase/decrease.

In Fig. 5, the left plot presents the estimated importance for channels 1 and 2 as a function of the angle theta. The estimation of relevance of the channels is expressive and captures how the importance shifts from channel 2 to channel 1 when the angle theta changes from 0° to 90°. Note that the relevance estimates made by the individual Detach-MiniRocket models present exhibit some variance, but the median of the distributions correctly captures the expected behavior. This shows how the ensemble nature of the model also helps to get a better estimate of the channel importance. In Fig. 5, the right plot shows the estimated relevance for all four channels when theta equals 45°. It can be observed that, even though the model assigns some relevance to the unimportant third and fourth channels, it clearly identifies that the first and second ones are the most important in an equal proportion.

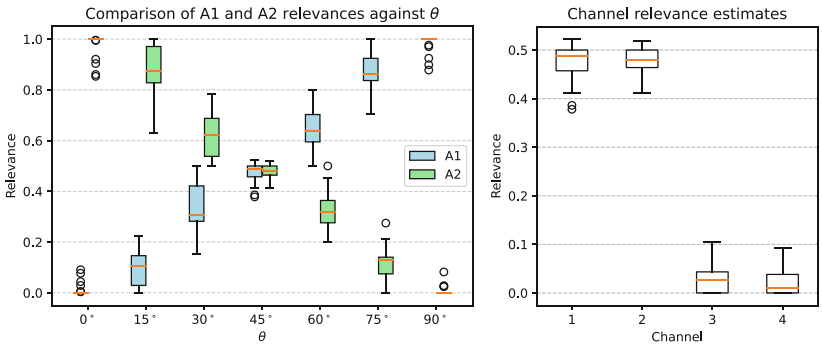


Fig. 5. Box plots of the synthetic dataset channel relevance estimates obtained with 25 Detach-MiniRocket models. The plot in the left shows the estimates of channels 1 and 2 (with sinusoidal amplitudes A1 and A2) along 7 different values of theta. For a given value of theta, box plots are presented side by side for better visibility. The right plot shows the estimates for all four channels when theta = 45°. Both figures are normalized so that the total relevance value adds up to one, and the orange lines indicate the median. (Color figure online)

This experiment demonstrates that the Detach-Rocket Ensemble provides an accurate estimate of the channels that are more relevant for classification

in this controlled environment. However, despite the effectiveness of our synthetic dataset in validating the methodology, it may not fully reflect the performance of the algorithm in complex real-world datasets, where factors such as high-dimensionality and low class separability may limit its ability to properly estimate channel relevance. In the following section, we illustrate the potential of our procedure in real-world neuroscience applications.

5 Real-World Datasets

5.1 Face Detection (MEG)

Dataset Description. We first tested our methodology on a MEG dataset collected and defined in [13] and shared in a Kaggle competition [10]. The data were acquired with an array containing 306 sensors: one magnetometer and two orthogonal planar gradiometers at each of the 102 positions across the scalp. The signals were sampled at 1.1 kHz and low-pass filtered with a cut-off frequency of 350 Hz. Using this configuration, data were collected from subjects who were presented with either pictures of faces or pictures of scrambled faces for a duration of less than one second. The proposed task consisted of classifying trials in which the subject observed a regular face against trials in which the subject was presented with a scrambled face.

Results. This dataset was used to test the performance of Detach-Rocket Ensemble in terms of both accuracy and channel relevance estimation. To evaluate its accuracy, we compared it to three other ROCKET-based models: MiniRocket, Detach-MiniRocket (abbreviated as D-MiniRocket), and Arsenal. To evaluate the channel relevance estimation, we compared it with previous analysis of relevant brain regions for the same task.

First, a hyperparameter optimization was conducted to determine the number of kernels that produced the most accurate results for MiniRocket, Detach-Rocket, and the Detach-Rocket Ensemble. To do this, 3 out of the 16 subjects present in the train set were used as a validation set on which the scores were computed, and the remaining 13 were used for training. The Appendix Table 2 shows the results for this exploration on all the models. The validation accuracy showed an overall increase with the number of kernels, capped at 10,000 kernels in the case of the Detach models, but beneficial up to the maximum number of explored kernels for MiniRocket (20,000). The best performing strategy in this initial search was a Detach-Rocket Ensemble with 10,000 kernels.

Given a specific dataset, further optimization of the Detach-Rocket Ensemble may improve the models performance. Some relevant parameters include the trade-off coefficient c, which tunes the extent to which the model is pruned, and the number of estimators N, which adds complexity to the ensemble. For this particular experiment we use $c = 0.1$, which is the default value for Detach-Rocket, and $N = 25$, to match the number of estimators of Arsenal's architecture.

We also evaluated the performance of Arsenal on the classification task, but we had to implement an alternative version using MiniRocket as base model instead of the default ROCKET. The reason for this was that the large number of channels present in the MEG dataset makes the default implementation unfeasible to run in our computational environment.

Finally, we compare the performance over three independent runs of the four models using the best configuration tested for each of them. The statistical results of these experiments are depicted in Fig. 6 (left). Detach-Rocket Ensemble obtains the best test accuracies among the evaluated models, while also showing lower overfitting than Arsenal as a consequence of the feature pruning. On average, the feature detachment process pruned the models down to 6.2% of their original features. In terms of average training time in our computational setting, a single MiniRocket model required 9.46 min, a Detach-MiniRocket model required 10.38 min, and the entire Detach ensemble required 263.66 min.

In Fig. 6 (right), we present the results of the channel relevance estimation using Detach-Rocket Ensemble. To evaluate our methodology, we compared the results with those obtained in the study from which the data originated [13]. In that study, the authors used statistical parametric mapping to find brain regions where MEG activity differed in recordings obtained from participants observing regular or scrambled faces. They found that the right lateral occipital cortex was the brain area that showed the most significant differences (see Fig. 6, panel A, for MEG data in [13]). This is precisely the same area that our methodology highlights as the most important for classification, demonstrating its ability to correctly estimate channel relevance in a complex high-dimensional real-world dataset.

Model	Train (%)	Test (%)
MiniRocket (20k kernles)	80.2±0.2	59.7±1.5
D-MiniRocket	72.2±2.9	60.8±0.5
Arsenal	87.4±0.1	61.5±0.4
D-Rocket Ensemble	**78.6± 0.3**	**64.3± 0.5**

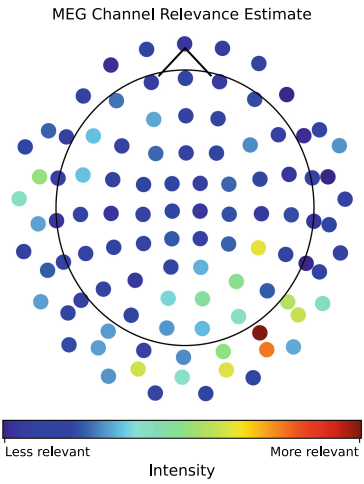


Fig. 6. (Left) Train and test accuracies (mean ± standard deviation) obtained by running the optimal models three times. (Right) Mean channel relevance estimates of the Face Detection task over three Detach-Rocket Ensembles.

5.2 Alzheimer's Disease Classification (EEG)

Dataset Description. This section employs the dataset presented in [19]. The dataset comprises EEG recordings from 88 participants. A total of 36 individuals were diagnosed with Alzheimer's disease (AD), 23 were diagnosed with frontotemporal dementia (FTD), and 29 were classified as healthy control subjects (CN). The mean age of each group was 66.4 years, 63.6 years, and 67.9 years, respectively. For each participant, the recording was conducted in a resting state with the eyes closed. The recording device utilized 19 scalp electrodes (Fp1, Fp2, F7, F3, Fz, F4, F8, T3, C3, Cz, C4, T4, T5, P3, Pz, P4, T6, O1, and O2) with two reference electrodes (A1 and A2) located on the mastoids. The preprocessing of these recordings included band-pass filtering within the range of 0.5 to 45 Hz and artifact correction using the Artifact Subspace Reconstruction routine (ASR) and Independent Component Analysis (ICA). The recording montage was referential using Cz for common mode rejection but, during the preprocessing, the signals were re-referenced to the average value of A1–A2.

Results. In order to evaluate the effectiveness of our model, we compare its performance with that of the study presented in [18]. The study, authored by the researchers who collected the dataset [19], introduces the Dual-Input Convolution Encoder Network (DICE-net), a classifier tailored to classify AD in this particular data modality. DICE-net employs engineered biomarkers to train a convolutional and transformer-based architecture. We conducted our experiments on the AD vs. CN classification task, as this is the task on which the study focuses.

To match the validation methodology used in [18], we employed Leave One Subject Out (LOSO) cross-validation, conducting the experiment 65 times, one fold for each of the subjects. This process yielded a set of 65 small confusion matrices—each obtained by training the model on 64 subjects and predicting the trials of the remaining one—which were then summed to derive the final results. In diagnostic tasks of this nature, a careful validation scheme, where trials in the test set belong solely to subjects unseen during training, is crucial for accurately evaluating the model's generalization ability. Table 1 presents the results, using the same evaluation metrics as in [18]. We split this table in two sections. The first shows models that require previous feature engineering, including DICE-net. The second part of this table shows the results for state-of-the-art deep learning models designed for raw EEG signal classification. Although not specifically designed for EEG, we include the Detach-Rocket Ensemble in this second part as it is also use raw EEG as input.

Table 1. AD vs CN scores reported by the models tested in [18] and our of Detach-Rocket Ensemble (bottom, D-Rocket Ensemble). The table includes both models requiring feature engineering and models using the raw EEG signal. The presented metrics are Accuracy (ACC), Sensitivity (SENS), Specificity (SPEC), Precision (PREC) and F1-Score (F1).

Type	AD/CN model	ACC	SENS	SPEC	PREC	F1
Feature engineering	LightGBM	76.28%	76.08%	76.52%	79.67%	77.83%
	XGBoost	75.53%	76.08%	74.87%	78.55%	77.29%
	CatBoost	75.39%	75.50%	75.25%	76.68%	77.05%
	SVM+PCA	73.75%	71.51%	76.46%	78.60%	74.89%
	PCA-kNN	72.52%	70.30%	75.19%	77.41%	73.69%
	MLP	73.69%	72.98%	74.81%	77.80%	75.31%
	DICE-net [18]	83.28%	79.81%	87.94%	88.94%	84.12%
Raw EEG	EEGNet [15]	41%	47.20%	37.67%	37.89%	42.04%
	EEGNetSSVEP [28]	51.46%	56.78%	45.39%	47.65%	51.82%
	DeepConvNet [23]	54.21%	45.43%	57.59%	48.71%	47.01%
	ShallowConvNet [23]	42.18%	46.50%	41.11%	49.74%	48.07%
	D-Rocket Ensemble	79.86%	78.89%	80.47%	74.89%	76.84%

Table 1 shows that our model significantly outperforms all alternatives using raw EEG data. These deep learning architectures designed for raw EEG signal classification overfit the training set and fail to generalize, achieving an accuracy no better than chance for unseen subjects. In fact, our model performs better than most of the models using feature engineering, being outperformed in accuracy only by DICE-net. Moreover, when implementing majority voting on subjects' trial predictions to obtain a class label per subject, our model achieves a subject-level accuracy of 86.15%, better than the 84.62% obtained by DICE-net (as inferred from Fig. 6 of [18]).

In addition to these results, we also present the Receiver Operating Characteristic (ROC) curve in Fig. 7 (left), obtained by sweeping different thresholds over the predicted class probabilities. We highlight two points on the curve. The first one corresponds to the scores initially obtained with the default threshold of 0.5. The second one shows the results obtained using the threshold that yielded the best accuracy, shown -along with the rest of the metrics- in Table 1, which is slightly better than the one obtained with the default threshold (79.86% instead of 79.80%). Note that this analysis is possible thanks to the existence of a probability value for the class labels.

Finally, with the LOSO procedure, we obtain 65 different estimations of channel importance for this classification task. In Fig. 7 (right), we present the estimated channel importance averaged over the 65 folds. This figure demonstrates the potential of our methodology: without any prior feature engineering,

we can identify the relevant brain areas for Alzheimer's disease classification in this dataset in a fully data-driven approach.

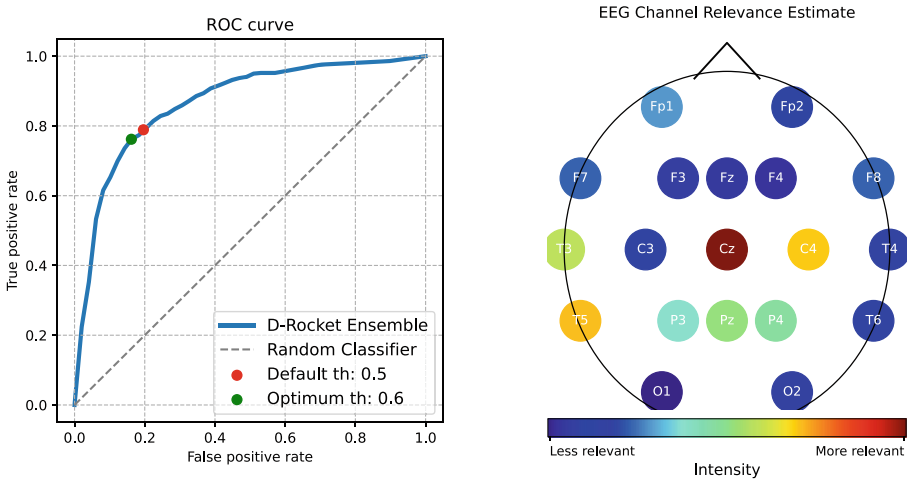


Fig. 7. (Left) Receiver Operating Characteristic (ROC) curve obtained by applying different thresholds (th) over the class probabilities of each trial. (Right) Channel relevance of the Alzheimer's Disease classification task estimated by the Detach-Rocket ensemble.

6 Discussion

Neuroscience applications typically involve multivariate time series data from a limited number of subjects with significant intersubject variability. This poses a challenge for deep learning classifiers, which can easily overfit the raw data and fail to generalize to subjects not included in the training set, creating a need for some form of feature engineering. In this context, simple ROCKET-based models have the potential to be an effective alternative, but they face some challenges regarding scalability with the number of channels and interpretability. This study aims to address these challenges by proposing Detach-Rocket Ensemble.

Similar to Arsenal, our proposed model is an ensemble of ROCKET-based models. As an ensemble, it has the advantages of reducing overfitting, being trivially parallelizable, and providing an intuitive label probability. However, the Detach-MiniRockets used in Detach-Rocket Ensemble are stronger based models, since they have a larger number of initial kernels and can better handle high-dimensional MTSC. In addition, after pruning, Detach-Rocket Ensemble ends up being a smaller model. For example, in the face detection challenge, the model was left with less than a third of the total kernels used by Arsenal.

One downside of the Detach-Rocket Ensemble is that the iterative detachment process must be conducted as many times as there are base models in the ensemble, creating an overhead in training time. However, this is compensated by the substantial reduction in the number of kernels during pruning, which makes the model require fewer convolutions during the forward pass, thus reducing inference time.

In future work, we plan to explore the iterative application of the Detach-Rocket Ensemble for channel pruning. After training the initial ensemble model, it is possible to use the estimated channel relevance to discard non-informative channels and then train a new ensemble model on the selected channels. This approach could improve model performance by providing a better coverage on the relevant channels. Additionally, it would be valuable to evaluate the Detach-Rocket Ensemble, using both Detach-MiniRocket and Detach-MultiRocket, on the UEA dataset to compare its performance with other non-ROCKET-based approaches.

7 Conclusion

In this study, we introduce Detach-Rocket Ensemble, an Multivariate Time Series Classification (MTSC) model that exploits the fast architecture of MiniRocket and the model size reduction provided by Sequential Feature Detachment (SFD) pruning. We demonstrate that Detach-Rocket Ensemble is able to handle both raw EEG and raw MEG data, achieving state-of-the-art performance while improving interpretability by providing built-in channel relevance and label probability.

Presented alongside a public repository with a user-friendly interface (https://github.com/gon-uri/detach_rocket), Detach-Rocket Ensemble represents a valuable resource for scientists working in the field of multivariate time series classification, particularly for neuroscience applications.

Acknowledgments. The authors would like to thank KTH Digital Futures and SciLifeLabs for granting their support in this project, as well as Federico Barone for engaging in insightful discussions.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

A Appendix

A.1 Face Detection Hyperparameter Optimization Tables

Table 2. Accuracies of several ROCKET variants on the validation set for different number of kernels. Both Detach-Rocket models use MiniRocket as the base model.

Model	Accuracy (%) for number of kernels			
	1000	5000	10000	20000
MiniRocket	57.45	58.30	61.83	62.05
Detach-Rocket (single model)	57.84	60.58	61.60	61.14
Detach-Rocket 10 model ensemble	59.10	61.83	**62.11**	61.66

References

1. Alzubaidi, L., et al.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J. Big Data **8**, 1–74 (2021)
2. Bagnall, A., et al.: The UEA multivariate time series classification archive. arXiv preprint arXiv:1811.00075 (2018)
3. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Disc. **31**, 606–660 (2017)
4. Britton, J.W., et al.: Electroencephalography (EEG): an introductory text and atlas of normal and abnormal findings in adults, children, and infants (2016)
5. Dau, H.A., et al.: The UCR time series archive. IEEE/CAA J. Autom. Sinica **6**(6), 1293–1305 (2019)
6. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Disc. **34**(5), 1454–1495 (2020)
7. Dempster, A., Schmidt, D.F., Webb, G.I.: MiniRocket: a very fast (almost) deterministic transform for time series classification. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 248–257 (2021)
8. Dempster, A., Schmidt, D.F., Webb, G.I.: Hydra: competing convolutional kernels for fast and accurate time series classification. Data Min. Knowl. Disc. **37**(5), 1779–1805 (2023)
9. Dietterich, T.G.: Ensemble methods in machine learning. In: International Workshop on Multiple Classifier Systems, pp. 1–15. Springer (2000)
10. Emanuele, Mosi, P.A.: DecMeg2014 - decoding the human brain (2014). https://kaggle.com/competitions/decoding-the-human-brain
11. Glover, G.H.: Overview of functional magnetic resonance imaging. Neurosurg. Clin. **22**(2), 133–139 (2011)
12. Hämäläinen, M., Hari, R., Ilmoniemi, R.J., Knuutila, J., Lounasmaa, O.V.: Magnetoencephalography-theory, instrumentation, and applications to noninvasive studies of the working human brain. Rev. Mod. Phys. **65**(2), 413 (1993)

13. Henson, R.N., Wakeman, D.G., Litvak, V., Friston, K.J.: A parametric empirical Bayesian framework for the EEG/MEG inverse problem: generative models for multi-subject and multi-modal integration. Front. Hum. Neurosci. **5**, 76 (2011)
14. Ismail Fawaz, H., et al.: InceptionTime: finding AlexNet for time series classification. Data Min. Knowl. Disc. **34**(6), 1936–1962 (2020)
15. Lawhern, V.J., Solon, A.J., Waytowich, N.R., Gordon, S.M., Hung, C.P., Lance, B.J.: EEGNet: a compact convolutional neural network for EEG-based brain-computer interfaces. J. Neural Eng. **15**(5), 056013 (2018)
16. Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., Bagnall, A.: HIVE-COTE 2.0: a new meta ensemble for time series classification. Mach. Learn. **110**(11–12), 3211–3243 (2021)
17. Middlehurst, M., Schäfer, P., Bagnall, A.: Bake off redux: a review and experimental evaluation of recent time series classification algorithms. arXiv preprint arXiv:2304.13029 (2023)
18. Miltiadous, A., Gionanidis, E., Tzimourta, K.D., Giannakeas, N., Tzallas, A.T.: Dice-net: a novel convolution-transformer architecture for Alzheimer detection in EEG signals. IEEE Access (2023)
19. Miltiadous, A., et al.: A dataset of scalp EEG recordings of Alzheimer's disease, frontotemporal dementia and healthy subjects from routine EEG. Data **8**(6) (2023). https://doi.org/10.3390/data8060095, https://www.mdpi.com/2306-5729/8/6/95
20. Mizrahi, D., Laufer, I., Zuckerman, I.: Comparative analysis of rocket-driven and classic EEG features in predicting attachment styles. BMC Psychol. **12**(1), 87 (2024)
21. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Disc. **35**(2), 401–449 (2021)
22. Rushbrooke, A., Tsigarides, J., Sami, S., Bagnall, A.: Time series classification of electroencephalography data. In: International Work-Conference on Artificial Neural Networks, pp. 601–613. Springer (2023)
23. Schirrmeister, R.T., et al.: Deep learning with convolutional neural networks for EEG decoding and visualization. Hum. Brain Mapp. **38**(11), 5391–5420 (2017)
24. Shifaz, A., Pelletier, C., Petitjean, F., Webb, G.I.: TS-chief: a scalable and accurate forest algorithm for time series classification. Data Min. Knowl. Disc. **34**(3), 742–775 (2020)
25. Tan, C.W., Dempster, A., Bergmeir, C., Webb, G.I.: MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. Data Min. Knowl. Disc. **36**(5), 1623–1646 (2022)
26. Uribarri, G., Barone, F., Ansuini, A., Fransén, E.: Detach-rocket: sequential feature selection for time series classification with random convolutional kernels. arXiv preprint arXiv:2309.14518 (2023). http://arxiv.org/abs/2309.14518
27. Uribarri, G., von Huth, S.E., Waldthaler, J., Svenningsson, P., Fransén, E.: Deep learning for time series classification of Parkinson's disease eye tracking data. arXiv preprint arXiv:2311.16381 (2023)
28. Waytowich, N., et al.: Compact convolutional neural networks for classification of asynchronous steady-state visual evoked potentials. J. Neural Eng. **15**(6), 066031 (2018)

Change Detection in Multivariate Data Streams: Online Analysis with Kernel-QuantTree

Michelangelo Olmo Nogara Notarianni[1(✉)], Filippo Leveni[1], Diego Stucchi[2], Luca Frittoli[1], and Giacomo Boracchi[1]

[1] Politecnico di Milano (DEIB), Milan, Italy
{michelangeloolmo.nogara,filippo.leveni,
luca.frittoli,giacomo.boracchi}@polimi.it
[2] STMicroelectronics, Agrate Brianza, Italy
diego.stucchi@st.com

Abstract. We present Kernel-QuantTree Exponentially Weighted Moving Average (KQT-EWMA), a non-parametric change-detection algorithm that combines the Kernel-QuantTree (KQT) histogram and the EWMA statistic to monitor multivariate data streams online. The resulting monitoring scheme is very *flexible*, since histograms can be used to model any stationary distribution, and *practical*, since the distribution of test statistics does not depend on the distribution of datastream in stationary conditions (non-parametric monitoring). KQT-EWMA enables controlling false alarms by operating at a pre-determined Average Run Length (ARL_0), which measures the expected number of stationary samples to be monitored before triggering a false alarm. The latter peculiarity is in contrast with most non-parametric change-detection tests, which rarely can control the ARL_0 a priori. Our experiments on synthetic and real-world datasets demonstrate that KQT-EWMA can control ARL_0 while achieving detection delays comparable to or lower than state-of-the-art methods designed to work in the same conditions.

Keywords: Online Change Detection · Non-parametric Monitoring · Multivariate Data Streams

1 Introduction

Change detection is a frequently faced challenge in data stream analysis, where the properties of some monitored process, e.g. a measurement acquired by a sensor, may change over time. In machine learning, changes in data distribution are known as concept drifts and pose challenges for classifiers and learning systems in general, requiring continuous adaptation.

In many application domains such as industrial monitoring, communication networks, and computer security, data come in virtually unlimited streams and need to be monitored *online*. In particular, each new observation needs to be processed immediately after being acquired, and must be evaluated considering the

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 115–130, 2025.
https://doi.org/10.1007/978-3-031-77066-1_7

whole data stream seen so far, while using a limited amount of memory and performing a fixed number of operations. In this study, we focus on online change-detection methods for multivariate data streams, which require algorithms capable of handling multidimensional vectors within these computational requirements and storage limitations. Another important challenge is posed by monitoring in a non-parametric manner, i.e., without any assumption on the initial data distribution. Non-parametric methods are particularly useful in real-world scenarios where the distribution of data is typically unknown. Unfortunately, most non-parametric change-detection algorithms are designed to monitor univariate data streams [15]. On top of that, controlling false alarms is a significant concern when each detected change can trigger costly interventions. Unfortunately, most online change-detection algorithms for multivariate data streams, particularly the non-parametric ones, struggle to effectively control false alarms. This paper addresses these challenges by proposing a method that is online, non-parametric, and capable of maintaining a target false alarm rate.

Online change detection monitoring techniques can be grouped into two categories: *one-shot* methods, which evaluate fixed-size batches of data points, and *sequential* methods, which do not require a fixed sample size and take into account the whole data stream. QuantTree (QT) is a *one-shot* non-parametric solution which, supported by theoretical results, guarantees a pre-set constant false positive rate (FPR). First presented in [2], QT algorithm defines a histogram, partitioning the d-dimensional input space. Non-parametric statistics can be computed over it, enabling change detection in multivariate data streams batch-wise. A fundamental limitation of QT is discussed in [12]: its splits are defined along the space axis, resulting in a hyper-rectangular partitioning that does not always adhere to the input distribution. To address this problem, a preprocessing stage is typically introduced to align the split directions to the principal components of the training set. However, it was observed [12] that this preprocessing can also worsen the control over false alarms. Hence, Kernel-QuantTree (KQT) was introduced, a generalized version of QT which partitions the space using kernel functions. The increased flexibility of the histogram in modeling the data distribution results in a *one-shot* monitoring of multivariate data streams with increased detection power. However, KQT is a batch-wise monitoring scheme using fixed-size windows and it fails to leverage the knowledge of the entire data stream distribution, thereby hindering fast detection of changes in an online scenario. A *sequential* version of QT algorithm, QT-EWMA, was presented in [4]. QT-EWMA computes the Exponentially Weighted Moving Average (EWMA) statistic on a QT histogram, thus considering the entire data stream acquired up to the current time instant t to monitor the data distribution. Moreover, QT-EWMA can control the average time elapsed before a false alarm is triggered (ARL_0). Although being a truly *sequential* extension of QT, QT-EWMA inherits the same weaknesses of its *one-shot* counterpart, i.e. the axis-parallel splits.

We propose Kernel-QuantTree Exponentially Weighted Moving Average (KQT-EWMA), a novel *sequential* non-parametric change-detection algorithm for multivariate data streams that extends KQT to the online scenario, following the approach of QT-EWMA. The theoretical properties of KQT guarantee that

KQT-EWMA is completely non-parametric since the distribution of our statistic does not depend on the data distribution, hence the thresholds controlling the ARL_0 can be set a priori, as in QT-EWMA. These thresholds guarantee by design a constant false alarm probability over time, thus a fixed false alarm rate at any time instant during monitoring.

Our extensive experimental analysis on both synthetic and real datasets shows that KQT-EWMA outperforms state-of-the-art existing methods, successfully extending KQT properties to the online scenario. Specifically, KQT-EWMA achieves control over the ARL_0 at a lower detection delay compared to competitors. We will show that, by relying on a precise partition of the space, KQT-EWMA outperforms QT-EWMA in complex scenarios, e.g., when analyzing data from multimodal distributions.

2 Problem Formulation

We consider a virtually unlimited multivariate data stream $x_1, x_2, \ldots$ in $\mathbb{R}^d$ where data samples x_t are i.i.d. realizations of a random variable with unknown distribution ϕ_0. We define the *change-point* $t = \tau$ as the unknown time instant when the distribution ϕ_0 experiences a change to ϕ_1, i.e.:

$$x_t \sim \begin{cases} \phi_0 & \text{if } t < \tau \\ \phi_1 & \text{if } t \geq \tau. \end{cases} \tag{1}$$

We further assume we are provided with a training set TR of N stationary realizations from ϕ_0, which is used to fit a model $\hat{\phi}_0$.

After estimating $\hat{\phi}_0$, online change-detection algorithms typically compute a statistic T_t at each observation x_t, to assess whether the new sequence $\{x_1, \ldots, x_t\}$ contains a change point or not. The decision rule usually involves checking whether $T_t > h_t$, where h_t is a given threshold. The detection time t^* is identified as the earliest time instant when sufficient statistical evidence indicates a change in the distribution, i.e.:

$$t^* = \min\{t : T_t > h_t\}. \tag{2}$$

A desirable property of change-detection algorithms is that the sequence of thresholds $\{h_t\}_t$ can be set *a priori* to guarantee a predefined $ARL_0 = \mathbb{E}_{\phi_0}[t^*]$, where the expectation is taken assuming that the whole data stream is drawn from ϕ_0. In practice, ARL_0, represents the expected time before a false alarm occurs, and plays a role similar to Type I error probability control in hypothesis testing. The change-detection goal is to detect a distribution change as soon as possible, thereby minimizing the detection delay $t^* - \tau$, while aiming for an empirical ARL_0 to approximate the predefined target value set beforehand.

3 Related Work

Change-detection algorithms are often *parametric* since they underpin hypotheses about the data distribution ϕ_0. As an example, the Change Point Model

(CPM) [15] based on the Hotelling test relies on the assumption that ϕ_0 conforms to Gaussian distribution. CPM performs online monitoring of data streams with theoretical guarantees regarding ARL_0 control. The CPM formulation can be extended to detect changes given an unknown non-Gaussian distribution when implemented with the Lepage statistic [10], always controlling false alarms. Unfortunately, the test statistics based on ranks are not suitable for multivariate scenarios. A *semi-parametric* change-detection strategy, Semi-Parametric Log-Likelihood (SPLL), was presented in [8]. SPLL models the initial data distribution ϕ_0 by fitting a Gaussian Mixture Model (GMM) $\hat{\phi}_0$ on a training set and then compares new incoming batches with batches from the training set using a likelihood test. Since SPLL does not provide a way to set the detection threshold a priori to control the ARL_0, it was combined with CPM [4]. In SPLL-CPM [4], SPLL reduces the dimensionality of incoming samples by computing their log-likelihood with respect to a GMM $\hat{\phi}_0$ fit on TR, thus the resulting univariate sequence can be conveniently monitored by a non-parametric extension of CPM leveraging the Lepage test statistic [10]. Again, the main limitation of both SPLL and SPLL-CPM is the assumption that ϕ_0 can be well approximated by a probability distribution of a known family (a GMM), which does not hold in general. There are only a few other multivariate methods that perform *non-parametric* change-detection. Scan-B [9] employs a Maximum Mean Discrepancy (MMD) statistic and can be configured to achieve a target ARL_0. However, the thresholds for this method are defined by analyzing the asymptotic behavior of ARL_0 when the size B of the sliding window is large [9]. Therefore, it fails at accurately controlling ARL_0. The NEWMA algorithm [7], also based on MMD, examines the relationship between two EWMA statistics with distinct forgetting factors. A limitation of this approach is that setting the ARL_0 thresholds requires the known analytical expression of ϕ_0. The Kernel-CUSUM [14] algorithm avoids assumptions about data distribution ϕ_0, but relies on a truncated approximation for ARL_0, which results in the underestimation of the thresholds [14]. QT [2] and QT-EWMA [4,5] are histogram-based change-detection methods featuring the desirable property that the distribution of test statistics, defined over bin probabilities, does not depend on the initial distribution ϕ_0. This allows to set detection thresholds a priori via Monte Carlo procedures, allowing for efficient false alarm control. KQT [12] defines histogram bins via nonlinear partition of the input space, resulting in a powerful change-detection algorithm in multivariate data streams. However, being a *one-shot* method, it cannot be directly employed for online change detection. Our proposal, namely KQT-EWMA, aims to extend KQT to the *sequential* scenario while retaining the capability of controlling false alarms given a target ARL_0 defined beforehand.

4 Kernel-QuantTree EWMA

We present KQT-EWMA, a novel change-detection algorithm which combines a KQT histogram [12], used as a model $\hat{\phi}_0$ of the stationary distribution ϕ_0, and the online statistic T_t based on an Exponential Weighted Moving Average [4]. In Sect. 4.1, we illustrate the KQT-EWMA algorithm, describing the

histogram construction, the threshold computation to control ARL_0, and the online monitoring process. In Sect. 4.2, we compare the computation complexity of KQT-EWMA against the alternatives considered in the experimental section. Finally, in Sect. 4.3 we discuss the limitations of KQT-EWMA.

4.1 The KQT-EWMA Algorithm

Algorithm 1 illustrates the training and inference phases of the KQT-EWMA algorithm. First, the KQT histogram $h = \{(S_k, \pi_k)\}_{k=1}^{K}$ is constructed over the training set $TR \subset \mathbb{R}^d$ to match a set of target probabilities $\{\pi_j\}_{j=1}^{K}$ (line 1), as described in [12]. The histogram construction process consists in iteratively splitting the input space into K bins defined as sub-level sets of a measurable function $\{f_k : \mathbb{R}^d \to \mathbb{R}\}_{k=1}^{K}$, which measures distances between training points and selected centroids using a kernel function. We remark that $K-1$ bins defined by KQT are compact subsets of $\mathbb{R}^d$. A sample that does not fall into any of these is assigned to the *residual* bin, which covers the unbounded remaining part of the space. We consider the Mahalanobis and the Weighted Mahalanobis (WM) distances as kernel functions, (as in [12]), but the properties of KQT hold for *any* measurable function projecting a multivariate vector in $\mathbb{R}^d$ on a single dimension.

As in [4], KQT-EWMA computes the weighted averages $\{Z_{j,t}\}$ (line 7), which keep track of the percentage of data stream samples $\{x_1, \dots, x_t\}$ falling in each bin S_j; to this purpose, we define K binary statistics $\{y_{j,t}\}_j$ as:

$$y_{j,t} = \mathbb{1}(x_t \in S_j), \tag{3}$$

for each $j \in \{1, \dots, K\}$ and $t \geq 1$ (line 6). As discussed in [4], under the assumption that the monitored samples $x_t \sim \phi_0$ are stationary, the expected values of the binary statistics in (3) can be approximated (line 2) as:

$$\mathbb{E}[y_{j,t}] \approx \hat{\pi}_j := \frac{N\,\pi_j}{N+1}, \quad j < K \quad \text{and} \quad \mathbb{E}[y_{K,t}] \approx \hat{\pi}_K := \frac{N\,\pi_K + 1}{N+1}. \tag{4}$$

During monitoring, each incoming sample x_t is used to update the weighted averages $\{Z_{j,t}\}$ and to compute the test statistic T_t. First, the sample is processed by the histogram h to obtain the binary statistics $\{y_{j,t}\}$ (line 6), which in turn are used to update the weighted averages $\{Z_{j,t}\}$ (line 7) as

$$Z_{j,t} = (1-\lambda)\, Z_{j,t-1} + \lambda\, y_{j,t} \quad \text{where} \quad Z_{j,0} = \hat{\pi}_j. \tag{5}$$

The past samples are weighted by an exponential curve which decreases with time constant λ. The expected value of the $Z_{j,t}$ statistic under ϕ_0 approximates $\hat{\pi}_j$, i.e. $\mathbb{E}[Z_{j,t}] \approx \hat{\pi}_j$ for $j = 1, ..., K$, thus the change-detection statistic is computed (line 8) as follows:

$$T_t = \sum_{j=1}^{K} \frac{(Z_{j,t} - \hat{\pi}_j)^2}{\hat{\pi}_j}. \tag{6}$$

Algorithm 1: KQT-EWMA

Input: training set $TR \subset \mathbb{R}^d$, target probabilities $\{\pi_j\}_{j=1}^K$, thresholds $\{h_t\}_t$, data stream to be monitored $x_1, x_2, \ldots, x_t, \cdots \subset \mathbb{R}^d$

Output: detection flag CD, detection time t^*

1 Construct the KQT histogram $\{S_j, \pi_j\}_{j=1}^K$ over TR as in [12]
2 Calculate the expected probabilities $\{\hat{\pi}_j\}_{j=1}^K$ as in (4)
3 Initialize the weighted averages $Z_{j,0} \leftarrow \hat{\pi}_j$ for each bin $j \in \{1, \ldots, K\}$
4 Initialize the detection flag $CD \leftarrow$ False and the detection time $t^* \leftarrow \infty$
5 **for** $t = 1 \ldots$ **do**
6 Compute the binary mask $y_{j,t} \leftarrow \mathbb{1}(x_t \in S_j)$
7 Update the random variables $Z_{j,t} \leftarrow (1 - \lambda)\, Z_{j,t-1} + \lambda\, y_{j,t},\ \forall j = 1 \ldots, K$
8 Compute the test statistic $T_t \leftarrow \sum_{j=1}^K (Z_{j,t} - \hat{\pi}_j)^2 / \hat{\pi}_j$
9 **if** $T_t > h_t$ **then**
10 $CD \leftarrow$ True, $t^* \leftarrow t$
11 **break**
12 **return** CD, t^*

The test statistic T_t measures the overall difference between the proportion of points in each bin S_j, represented by $Z_{j,t}$, and their approximated expected values $\hat{\pi}_j$ under ϕ_0, thus corresponds to the Pearson statistic. The statistic naturally increases as a consequence of a change $\phi_0 \to \phi_1$ that modifies the probability of at least one bin. Finally, the statistic T_t is compared against the corresponding threshold h_t to detect a change (line 10).

False Alarms and Threshold Computation Strategy. KQT-EWMA algorithm inherits from Kernel-QuantTree the fundamental property that the distribution of the statistics in (6) does not depend on the data distribution ϕ_0. The true bin probabilities $p_j = \mathbb{P}_{\phi_0}(S_j)$, i.e. the set of probabilities of a point sampled from ϕ_0 to belong to the bin S_j, are drawn from the Dirichlet distribution $(p_1, \ldots, p_K) \sim \mathcal{D}(\pi_1 N, \pi_2 N, \ldots, \pi_K N + 1)$ where $\{\pi_j\}_{j=1}^K$ is the set of target probabilities, as demonstrated in [12]. It follows that the distribution of any statistic based on KQT, including T_t, does not depend on ϕ_0 [2,4]. Thus, the thresholds $\{h_t\}_t$ can be defined a priori to control ARL_0 on any data stream, which is defined as:

$$ARL_0 = \mathbb{E}_{\phi_0}[t^*] = \frac{1}{\alpha}. \tag{7}$$

As explained in [5], detection thresholds h_t guarantee the constant false alarm probability, i.e. thresholds are such that the following equation is satisfied:

$$\mathbb{P}(T_t > h_t \mid T_k \leq h_k\ \forall k < t) = \alpha \quad \forall t \geq 1. \tag{8}$$

Since detection time t^* is a Geometric random variable with parameter α, the probability of encountering a false alarm before time t can be determined through the geometric sum:

$$\mathbb{P}(t^* \leq t) = \sum_{k=1}^{t} \alpha(1-\alpha)^{k-1} = \alpha \cdot \frac{1-(1-\alpha)^t}{\alpha} = 1-(1-\alpha)^t. \tag{9}$$

Thus, we can monitor the control over false alarms in data streams containing a change point τ by computing the proportion of streams in which $t^* \leq \tau$.

We leverage the results in [5] which proves that, to estimate the thresholds h_t, one can directly simulate the construction of QT histograms on a training set $TR \sim \phi_0$ of size N by drawing its bin probabilities from the Dirichlet distribution, $(p_1, \ldots, p_K) \sim \mathcal{D}(\pi_1 N, \pi_2 N, \ldots, \pi_K N + 1)$. This approach holds with KQT given any kernel function, i.e. we can use the same threshold sequences given any measurable function, including linear split functions along the axis, which would result in a QT histogram. Therefore, the same thresholds, computed in a Monte Carlo scheme, can be used for QT-EWMA and KQT-EWMA to guarantee a constant false alarm probability over time. The thresholds do not depend on the data distribution ϕ_0 nor the data dimension d. The entire simulation procedure must be repeated when changing the λ parameter of the EWMA statistic, the target bin probabilities $\{\pi_j\}_{j=1}^{K}$, or the training set size N.

4.2 Computational Complexity

Since efficiency is key in online monitoring, we analyze the computational complexity of KQT-EWMA in comparison with QT, QT-EWMA, KQT, and SPLL, SPLL-CPM, and Scan-B. The results are summarized in Table 1. Further explanations can be found in [5,12].

The training of a KQT given a training set TR of N points comprises i) the projection of TR by f_k, whose cost depends on the specific kernel function, ii) the computation of the split value, which costs $\mathcal{O}(N)$, and iii) the centroid selection. The cost of computing the Euclidean distance - or other distances based on l_p norms - is $\mathcal{O}(d)$, while the Mahalanobis distance costs $\mathcal{O}(d^2)$ and the Weighted Mahalanobis (WM) distance costs $\mathcal{O}(M\,d^2)$, where M is the number of Gaussian components fitted to TR and d is the data dimension. The centroid selection criteria is based on the information gain, which estimate is dominated by the computation of the determinant of the sample covariance matrix, which costs $\mathcal{O}(d^3)$. Overall, the cost of the index computation is multiplied by the number of centroids V tested during the selection procedure; therefore, an upper bound for the cost of KQT construction is $\mathcal{O}(K\,V\,(N + M\,N\,d^2 + d^3))$ when using the WM distance and the information gain criteria. During monitoring, the only operation performed is the projection by f_k of the samples, resulting in a cost of $\mathcal{O}(K\,M\,d^2)$ in case of the WM distance.

4.3 Discussion and Limitations

The main limitation of KQT-EWMA is that it is based on measures requiring the computation of the sample covariance matrix, which can be challenging in high-dimensional data streams. In KQT, given any kernel function, the centroid

Table 1. Training and inference costs of KQT-EWMA with Weighted Mahalanobis (WM) distance and distances derived from l_p norms (e.g. Euclidean distance when $p = 2$), compared against the other considered methods. V is the number of centroids tested to build each bin, M is the number of Gaussian components fit on the dataset, K is the number of bins, and N is the training set size. As for the other methods, m is the number of Gaussian components and w is the window length used by SPLL; n is the number of windows of B samples employed by Scan-B.

Method	Training Cost	Inference Cost (per sample)
KQT-EWMA (WM)	$O(K\,V(N + M\,N\,d^2 + d^3))$	$O(K\,M\,d^2)$
KQT-EWMA (l_p)	$O(K\,V(N + N\,d + d^3))$	$O(K\,d)$
QT-EWMA	$O(K\,N\,\log N)$	$O(K)$
SPLL (online)	$O(m\,N\,d^2)$	$O(m\,d + w\,\log w)$
Scan-B	N.A.	$O(n\,B\,d)$

selection criteria is the maximum information gain: the best split lowers the data entropy [12], which is computed as $H(B) = (1/2)\log\left((2\pi e)^d \det(\mathrm{cov}[B])\right)$, where $\mathrm{cov}[B]$ is the sample covariance matrix computed over a set of points B. Moreover, the sample covariance matrix estimated from the training set TR is used to define the Mahalanobis and the WM distances. The problem of determining the minimal sample size N that guarantees that the sample covariance matrix approximates the actual covariance matrix depends on the data distribution, as well explained in [13]. Our experiments shows that KQT-EWMA can lose control over ARL_0 when few training points are provided.

QT-EWMA-update Algorithm [5] is an effective monitoring scheme when N is relatively small, i.e. when there are a few training samples, as this estimates the bin probabilities incrementally as new observations are available, as long as no changes are detected. While an incremental variant of KQT-EWMA can be implemented, this would be impractical due to computational and memory requirements, as it would require re-computing covariance matrices and centroids (possibly in a high dimensional space) at each update.

5 Experiments

The goal of our experiments is to show that KQT-EWMA controls the false alarms while achieving state-of-the-art detection delays. To do this, we will show empirical results obtained on both synthetic and real-world data streams. In KQT-EWMA, as in QT-EWMA [4], we set the number of bins to $K = 32$ and uniform target probabilities $\pi_j = 1/K$. The exponential decay of the EWMA statistic is given by a time constant $\lambda = 0.05$. To monitor with QT and SPLL we set the batch size $\nu = 32$ as in [4], and we employ the original configuration of the Scan-B algorithm [9] ($n = 5$ windows of $B = 100$ samples), if not specified otherwise. We set window length $w = 1000$ for SPLL-CPM. The number of centroids tested to build each bin of KQT-EWMA is $V = 250$.

5.1 Datasets

Synthetic: As in [5], we generate synthetic data streams in spaces of increasing dimension $d \in \{2, 4, 8, 16, 32, 64\}$. We use Gaussian distributions ϕ_0 with a random covariance matrix, and then we define the post-change distribution $\phi_1 = \phi_0(Q + v)$ as a random roto-translation of ϕ_0. The roto-translation parameters Q and v are generated using the CCM framework [3] to guarantee a fixed distance between the two distributions computed as the symmetric Kullback-Leibler divergence $sKL(\phi_0, \phi_1) \in \{0.5, 1, 1.5, 2, 2.5, 3\}$. We expand our analysis to datasets sampled from bi-modal and tri-modal Gaussians to show the benefits of the distribution estimation with a KQT histogram for change detection.

Real-World: As in [4], we test seven multivariate classification datasets of varying dimensionality: El Niño Southern Oscillation ("niño", $d = 5$), Physicochemical Properties of Protein Ternary Structure ("protein", $d = 9$), two of the Forest Covertype datasets ("spruce" and "lodgepole", $d = 10$), Credit Card Fraud Detection ("credit", $d = 28$), Sensorless Drive Diagnosis ("sensorless", $d = 48$), and MiniBooNE particle identification ("particle", $d = 50$). We preprocess these datasets from the UCI Machine Learning Repository [6] as in [4]: "particle", "protein", "credit", and "sensorless" datasets are standardized with respect to the standard score, and we sum to each component of "sensorless", "particle", "spruce" and "lodgepole" imperceptible Gaussian noise to avoid repeated values, which harm the construction of QT histograms. The distributions of these datasets are considered to be stationary [4]. We randomly sample the data streams and introduce changes $\phi_0 \to \phi_1$ by shifting the each distribution by a random vector drawn from a d-dimensional Gaussian scaled by the total variance of the dataset. We show the analysis of UCI datasets as the average results obtained over all the datasets.

Our experiments also include the INSECTS dataset [11], which contains $d = 33$ attributes derived from the wing-beat frequency of various insects, captured via an optical sensor. This dataset, tailor-made for change-detection techniques, contains records under diverse environmental conditions impacting insect flight behaviors. We focus on the abrupt-change variant of this dataset, which includes five distinct distribution changes $\phi_0 \to \phi_1 \to \cdots \to \phi_5$. We sample data points from these distributions to build our training set TR and test data streams. Results obtained over the five changes present in the INSECTS datasets are averaged and shown all together.

5.2 Figures of Merit

Empirical $\mathbf{ARL_0}$. To assess whether KQT-EWMA and the other considered methods control the target ARL_0 (see (7)), we compute its empirical value as the average time before raising a false alarm on data streams we sample from ϕ_0. Empirical ARL_0 values are measured on 4000 data streams drawn from ϕ_0, given a target ARL_0 taking values in {500, 1000, 2000, 5000}. We generate stationary data streams of length $L = 6 \cdot ARL_0$ - the corresponding probability to detect a false alarm in each sequence is thus $\mathbb{P}(t^* \leq L) \approx 0.9975$, as in [4].

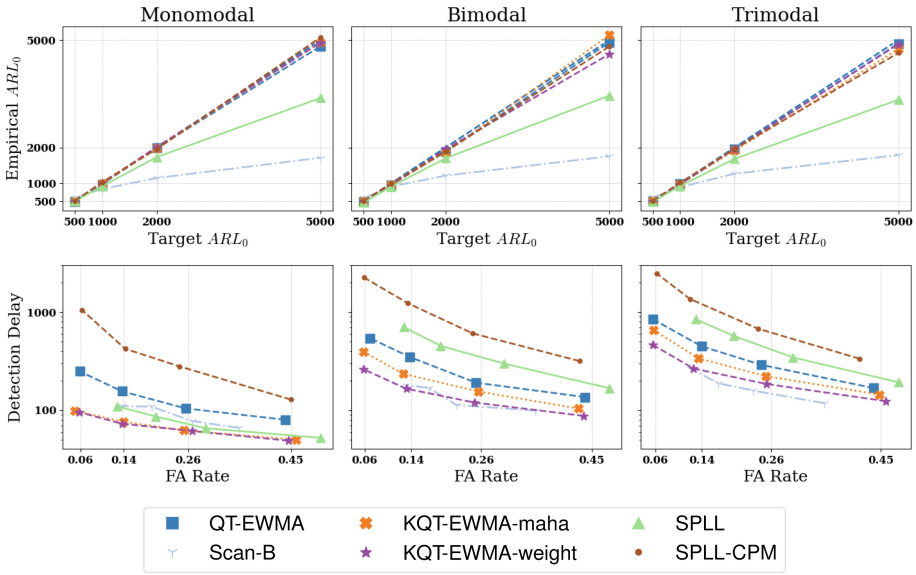


Fig. 1. Empirical ARL_0 and detection delay achieved by the considered methods monitoring data streams generated by Gaussian mixtures with increasing number of components (1, 2, 3). We show that as the number of components increases, KQT-EWMA with Weighted Mahalanobis (WM) distance advantage in terms of detection delay increases, achieving in general the lowest delays while controlling false alarms. In all the experiments, the GMM used to compute the WM distance fits $M = 4$ components.

Detection Delay. We evaluate the detection power of KQT-EWMA and the other considered methods by their detection delay, i.e., $ARL_1 = \mathbb{E}[t^* - \tau]$, where the expectation is taken assuming that a change point τ is present. Again, we set the target ARL_0 a priori, $ARL_0 \in \{500, 1000, 2000, 5000\}$. Results are averaged over 4000 data streams of length $6 \cdot ARL_0$, each containing a change point at $\tau = 300$. This is a difference compared to the analysis in [4,5], where the average detection delay is computed at any given target ARL_0 on sequences of fixed length. We use sequences of the same length to estimate detection delay and ARL_0 to achieve a fair comparison between these two quantities.

False Alarm Rate. The False Alarms (FA) rate is computed as the number of alarms raised at some $t < \tau$, averaged over 4000 experiments. By setting the target ARL_0 to $\{500, 1000, 2000, 5000\}$, we expect the percentage of false alarms to be $\{45\%, 26\%, 14\%, 6\%\}$, respectively, as stated by (9). The target FA rates are indicated in the plots by vertical dotted lines.

5.3 Results and Discussion

False Alarms Control. To show the control over false alarms, we plot the empirical ARL_0 obtained in our experiments against the target ARL_0 set a

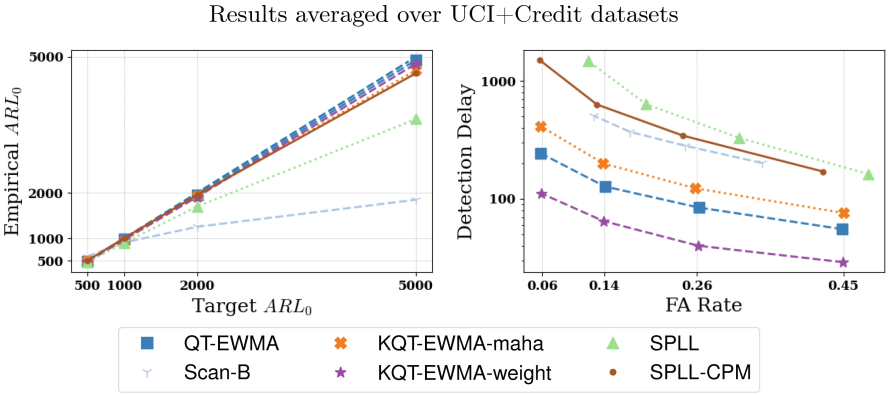


Fig. 2. Average empirical ARL_0 and detection delay on data streams sampled from the UCI datasets, excluding the highest-dimensional ones (i.e., "particle" and "sensorless"). In these two cases, $N = 4096$ training samples are not enough for KQT-EWMA based on Mahalanobis and WM distances to properly control ARL_0. In this setting, KQT-EWMA with WM distance achieves by far the best performance, halving the detection delay of QT-EWMA while controlling the target ARL_0.

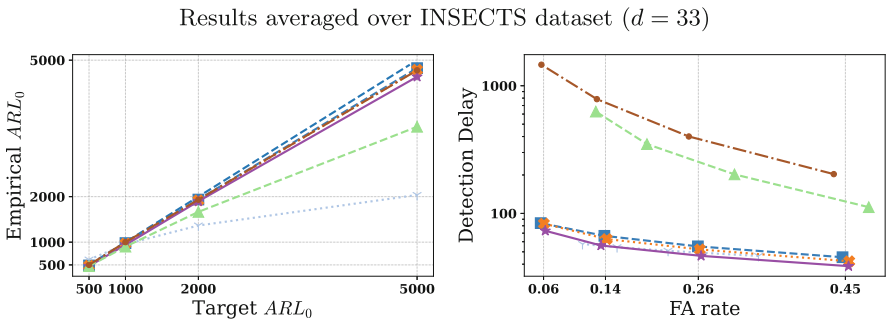


Fig. 3. Average empirical ARL_0 and detection delay on data streams from the INSECTS dataset [11], with different combinations of ϕ_0 and post-change distribution ϕ_1. QT-EWMA and KQT-EWMA achieve similar detection delays, while KQT-EWMA with WM distance struggles in controlling higher values of ARL_0.

priori. We compare results obtained over data sampled from monomodal Gaussians with the same performance measures computed over multimodal Gaussian datasets (bimodal and trimodal, in Fig. 1, first row). In all the experiments, the number of components used to fit the GMM and compute the WM distance is $M = 4$. QT-EWMA and SPLL-CPM can control all the target values chosen for ARL_0, while in general SPLL and Scan-B struggle in achieving high target $ARL_0 \in \{2000, 5000\}$. This is true also considering the results obtained with others real-world data sets (see Figs. 2 and 3).

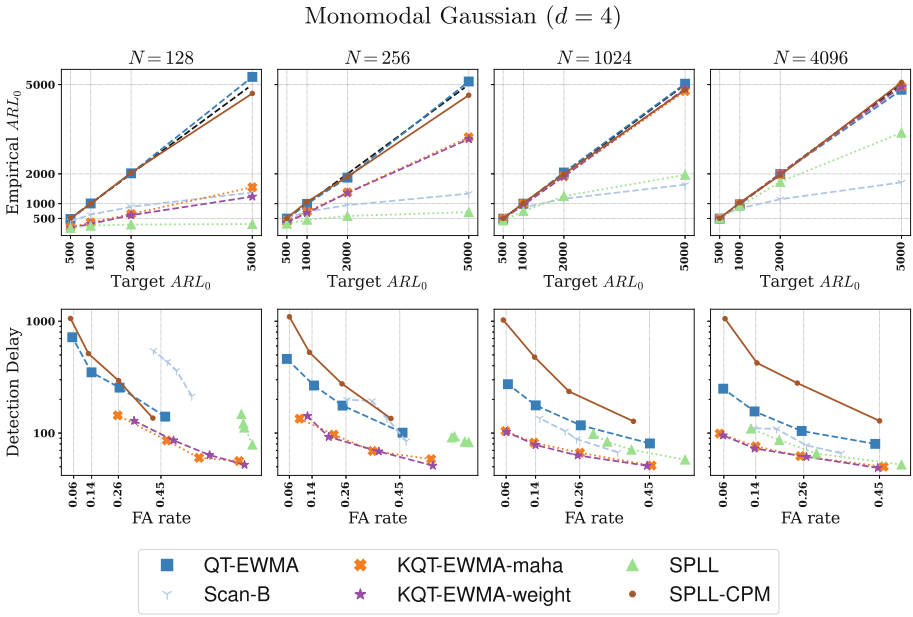


Fig. 4. Empirical ARL_0 and detection delay on Gaussian data streams in $d = 4$ dimensions, for varying training set sizes $N \in \{128, 256, 1024, 4096\}$. The empirical ARL_0 (first row) of QT-EWMA and SPLL-CPM always approaches the target values $(500, 1000, 2000, 5000)$, while the other methods cannot control the ARL_0. When the training set size N is sufficiently large ($N \in \{1024, 4096\}$), KQT-EWMA can control the FA rate, and achieves the lowest detection delay when using the Mahalanobis or the WM distance.

KQT-EWMA effectively controls target values of ARL_0, while achieving the lowest detection delays in these scenarios(see Fig. 1, second row). Figure 2 (first row) shows experimental results averaged over data streams sampled from 5 UCI datasets having $d \leq 28$, for which we used $N = 4096$ training points. We show two high-dimensional datasets ("particle" and "sensorless", $d = 50$ and $d = 48$ respectively, see Fig. 6) separately, since 4096 training points are not enough to estimate the sample covariance matrix in high dimensions, and KQT-EWMA based on Mahalanobis and WM distances do not properly control the ARL_0. In particular, KQT-EWMA with WM distance achieves low empirical ARL_0 values due to the difficulty in fitting a GMM here.

Figure 7 plots the change magnitude $sKL(\phi_0, \phi_1)$ against the detection delay achieved by the considered methods monitoring Gaussian data sequences with a target $ARL_0 = 1000$. While all methods successfully control the false alarms, KQT-EWMA achieves the lowest detection delay, even in this setting where the parametric assumptions of SPLL are met (number of Gaussian components is set to $m = 1$). The advantage of KQT-EWMA over the alternatives is especially noticeable when the divergence between the pre- and post-change distributions

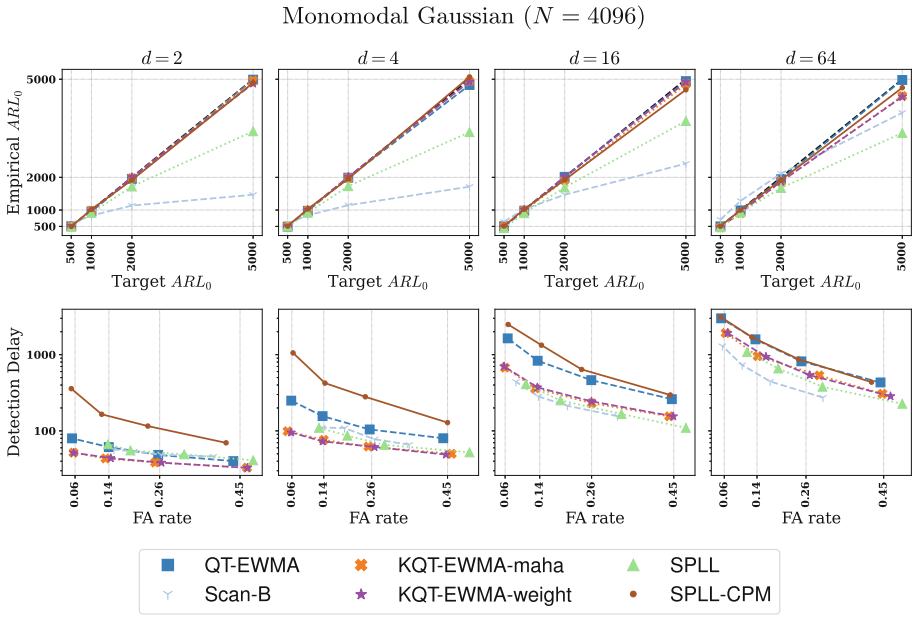


Fig. 5. Empirical ARL_0 and detection delay on data streams drawn from a Gaussian distribution with dimension $d \in \{2, 4, 16, 64\}$, trained over $N = 4096$ stationary samples. The empirical ARL_0 (first row) of KQT-EWMA, QT-EWMA, and SPLL-CPM always match the target, while Scan-B and SPLL fail. However, KQT-EWMA using the Mahalanobis distance cannot control the ARL_0 well when $d = 64$, as $N = 4096$ training points are not sufficient to estimate such a high-dimensional covariance matrix. When $d \leq 16$, the detection delay (second row) achieved with KQT-EWMA with Mahalanobis distance is the lowest achieved among the methods controlling the ARL_0.

is low ($sKL = 0.5$). As expected, the detection delay of all methods decreases when the change magnitude increases.

Detection Delay vs False Alarms. To assess the detection power of these models, we plot the average detection delay against the percentage of false alarms. Figure 1 (second row) shows that KQT-EWMA with Mahalanobis and WM distances achieves the lowest detection delay regardless of the number of modalities, alongside Scan-B. However, Scan-B is unable to control higher values of ARL_0. Similarly, SPLL cannot control ARL_0, and the distance from the target values is even more pronounced. This is also evident in the results on real data (Figs. 2 and 3). If we compare KQT-EWMA to QT-EWMA, which has the second-best detection delay values in the Gaussian scenario (Fig. 1, second row), we can observe that KQT-EWMA more than halves the detection delay of QT-EWMA. Moreover, this difference increases as the complexity of the underlying distribution rises. This result is confirmed by all the experiments on both real (Figs. 2 and 3) and synthetic (Figs. 4 and 5) datasets, showing that the histogram construction strategy of KQT-EWMA, coupled with the Maha-

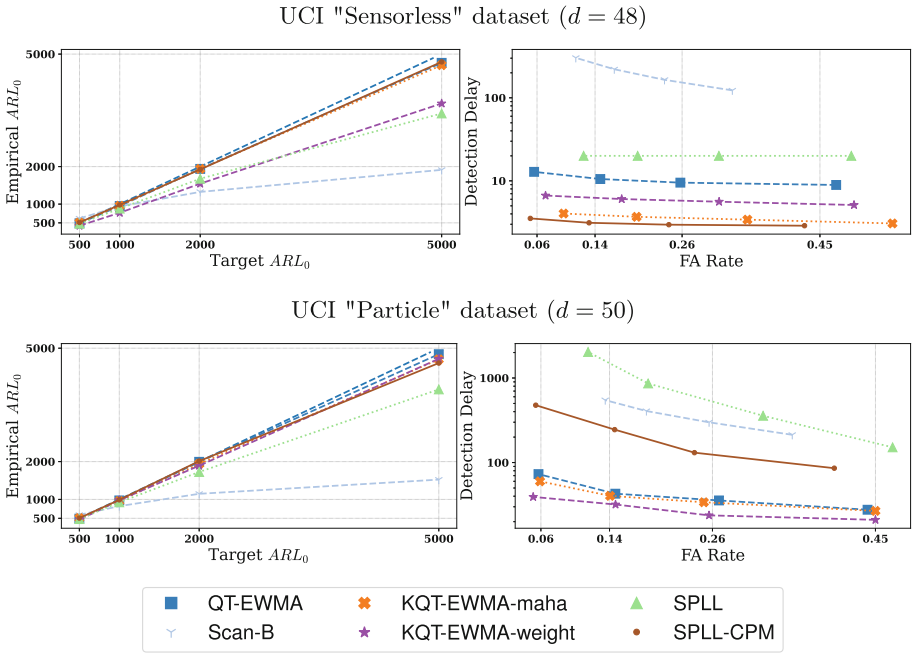


Fig. 6. Empirical ARL_0 and detection delay achieved by the considered methods monitoring the two high-dimensional UCI data sets "particle" ($d = 48$, above) and "sensorless" ($d = 50$, below). The $N = 4096$ training samples used in these experiments are not enough for KQT-EWMA based on Mahalanobis and WM distances to properly control ARL_0. Results are averaged over 4000 experiments.

lanobis and WM distances, improves the detection performance, achieving lower detection delays. Figure 5 shows the effects of *detectability loss* [1]: the ability to perceive a distribution change diminishes as the data dimensionality d increases while the distance between pre- and post-change distribution is fixed ($sKL = 1$), thus detection delays increase.

Figure 7 illustrates the relation between the detection delay and the change magnitude between pre- and post-change Gaussian sequences. We set target $ARL_0 = 1000$ for all methods. All methods successfully control the false alarms, and KQT-EWMA achieves the lowest detection delay, even when the parametric assumptions of SPLL are met (number of Gaussian components is set to $m = 1$). The advantage of KQT-EWMA over the alternatives is especially noticeable when the divergence between the pre- and post-change distributions is low ($sKL = 0.5$). As expected, the detection delay of all methods decreases when the change magnitude increases.

Our extensive analysis shows that KQT-EWMA consistently achieves the lowest detection delays across different scenarios. Overall, KQT-EWMA based on Mahalanobis and WM distances can detect distribution changes more effectively, especially when the complexity of the underlying distribution rises.

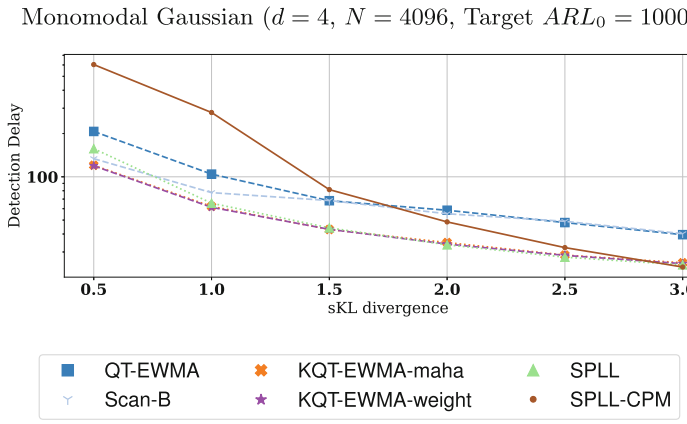


Fig. 7. Detection delay as a function of the magnitude of the change $\phi_0 \rightarrow \phi_1$ between pre- and post-change Gaussian sequences. We set the target $ARL_0 = 1000$. KQT-EWMA achieves the lowest detection delay, even in the challenging scenario when the change magnitude is low ($sKL = 0.5$). As expected, all methods decrease their detection delays when the change magnitude increases. We remark that the empirical ARL_0 achieved by SPLL and Scan-B is lower than the target.

6 Conclusion and Future Works

We introduce KQT-EWMA, a non-parametric online change-detection algorithm for multivariate data streams based on Kernel-QuantTree [12]. The theoretical results underpinning KQT-EWMA [5,12] guarantee the control of false alarms independently on the initial data distribution. Our experiments on synthetic and real-world data streams show that KQT-EWMA achieves state-of-the-art detection delay while effectively controlling false alarms.

In particular, the algorithm can leverage any measurable kernel function and it is able to fit complex distributions, resulting in high detection power. The sequences of thresholds can be computed independently on the data distribution ϕ_0, the data dimension d, and the selected kernel function. Moreover, the monitoring scheme is invariant to roto-translation of the input data (when employing Mahalanobis and Weighted Mahalanobis distances, as shown in [12]), thus KQT-EWMA does not require any preprocessing step such as PCA.

Our experimental evaluation also delineates some limitations: while the computational complexity of QT-EWMA scales well with the data dimension d during both training and testing phases, KQT-EWMA's computational complexity does not, potentially impacting its practical utility in high-dimensional scenarios. Additionally, KQT-EWMA relies on the sample covariance matrix, whose estimation can be poor in high-dimensional scenarios where the training set TR is not sufficiently large. Nevertheless, our experiments show that KQT-EWMA achieves excellent performance compared to the other methods designed for online monitoring, including QT-EWMA, while effectively con-

trolling the false alarms, especially when considering complex data distributions such as multi-modal Gaussians or real-world datasets.

Future work concerns addressing the limitations of KQT-EWMA with high-dimensional datasets. Specifically, we plan to design kernels that do not rely on covariance matrix computation and are specifically tailored for the *sequential* high-throughput scenario.

References

1. Alippi, C., Boracchi, G., Carrera, D., Roveri, M.: Change detection in multivariate datastreams: likelihood and detectability loss. In: International Joint Conference on Artificial Intelligence (IJCAI), vol. 2, pp. 1368–1374 (2016)
2. Boracchi, G., Carrera, D., Cervellera, C., Macciò, D.: QuantTree: histograms for change detection in multivariate data streams. In: Proceedings of the 35th International Conference on Machine Learning, vol. 80, pp. 639–648. PMLR (2018)
3. Carrera, D., Boracchi, G.: Generating high-dimensional datastreams for change detection. Big Data Res. **11**, 11–21 (2018)
4. Frittoli, L., Carrera, D., Boracchi, G.: Change detection in multivariate datastreams controlling false alarms. In: Machine Learning and Knowledge Discovery in Databases. Research Track, pp. 421–436. Springer, Cham (2021)
5. Frittoli, L., Carrera, D., Boracchi, G.: Nonparametric and online change detection in multivariate datastreams using QuantTree. IEEE Trans. Knowl. Data Eng. **25**(8), 8328–8342 (2022)
6. Kelly, M., Longjohn, R., Nottingham, K.: The UCI machine learning repository. https://archive.ics.uci.edu
7. Keriven, N., Garreau, D., Poli, I.: NEWMA: a new method for scalable model-free online change-point detection. IEEE Trans. Signal Process. **68**, 3515–3528 (2020). https://doi.org/10.1109/TSP.2020.2990597
8. Kuncheva, L.I.: Change detection in streaming multivariate data using likelihood detectors. IEEE Trans. Knowl. Data Eng. **25**(5), 1175–1180 (2013)
9. Li, S., Xie, Y., Dai, H., Song, L.: Scan B-statistic for kernel change-point detection. Seq. Anal. **38**(4), 503–544 (2019)
10. Ross, G.J., Tasoulis, D.K., Adams, N.M.: Nonparametric monitoring of data streams for changes in location and scale. Technometrics **53**(4), 379–389 (2011)
11. Souza, V.M.A., dos Reis, D.M., Maletzke, A.G., Batista, G.E.A.P.A.: Challenges in benchmarking stream learning algorithms with real-world data. Data Min. Knowl. Discov. **34**, 1805–1858 (2020)
12. Stucchi, D., Rizzo, P., Folloni, N., Boracchi, G.: Kernel QuantTree. In: Proceedings of the 40th International Conference on Machine Learning (2023)
13. Vershynin, R.: How close is the sample covariance matrix to the actual covariance matrix? J. Theor. Probab. **25** (2010). https://doi.org/10.1007/s10959-010-0338-z
14. Wei, S., Xie, Y.: Online kernel CUSUM for change-point detection (2022). https://doi.org/10.48550/arXiv.2211.15070
15. Zamba, K.D., Hawkins, D.M.: A multivariate change-point model for statistical process control. Technometrics **48**(4), 539–549 (2006)

Weighted Average of Human Motion Sequences for Improving Rehabilitation Assessment

Ali Ismail-Fawaz[1(✉)], Maxime Devanne[1], Stefano Berretti[2], Jonathan Weber[1], and Germain Forestier[1,3]

[1] IRIMAS, Université de Haute-Alsace, Mulhouse, France
{ali-el-hadi.ismail-fawaz,maxime.devanne,jonathan.weber, germain.forestier}@uha.fr
[2] MICC, University of Florence, Florence, Italy
stefano.berretti@unifi.it
[3] DSAI, Monash University, Melbourne, Australia
germain.forestier@monash.edu

Abstract. While human motion analysis has been widely addressed in recent years, the specific task of rehabilitation motion assessment remains challenging due to the lack of available annotated data. To overcome this challenge, data augmentation can be considered. However, classical augmentation techniques applied to human motion sequences often result in meaningless movements. Moreover, in rehabilitation assessment, labels are often continuous values illustrating the quality of a movement. Hence, associating a continuous label to augmented data is not straightforward. In this work, we propose to address data augmentation using an averaging method, called *shapeDBA*, adapted to rehabilitation motion sequences represented as multivariate time series. We extend the original proposal by weighting the average, hence allowing us to infer continuous labels associated to augmented motion sequences. We evaluated our proposed method on the Kimore dataset. Experimental results show that our method generates coherent rehabilitation sequences that can be efficiently used to extend a small dataset for rehabilitation assessment.

Keywords: Rehabilitation assessment · Data extension · Synthetic motion data

1 Introduction

Human motion data is everywhere in our daily lives and has important applications in the medical sector such as in human rehabilitation assessment. It helps doctors and therapists to track patient progress, design personalized recovery plans and improve treatment outcomes. To capture human movement, while expensive and accurate Motion Capture (MoCap) systems are mainly used in the entertainment domain, they may be unsuitable for rehabilitation as they require

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, pp. 131–146, 2025.
https://doi.org/10.1007/978-3-031-77066-1_8

the wearing of sensors or dedicated suits. Conversely, skeleton data extracted from videos are much less intrusive, and thus are increasingly adopted in rehabilitation centers.

In rehabilitation assessment, a challenging task involves analysing a rehabilitation exercise to predict a performance score corresponding to how well it has been performed by a patient. Performance scores are continuous values often ranging from 0 to 100, hence estimating such scores can be modeled as an extrinsic regression problem. Extrinsic regression refers to predicting an output continuous variable based on input data samples using machine learning models [8].

Deep learning approaches have been successfully employed for various skeleton-based human motion analysis tasks, where consequent datasets are available. However, such approaches may be limited when a small amount of motion sequences is available, like in the medical field. This is the case for rehabilitation assessment, where acquiring real motion sequences performed by patients during their rehabilitation program is not straightforward [1]. The data collection task is further complicated when it comes to annotations, as associating a performance score to each acquired rehabilitation sequence is time consuming and requires the support of clinical experts. These reasons certainly explain the few amount of publicly available rehabilitation datasets as well as their limited size.

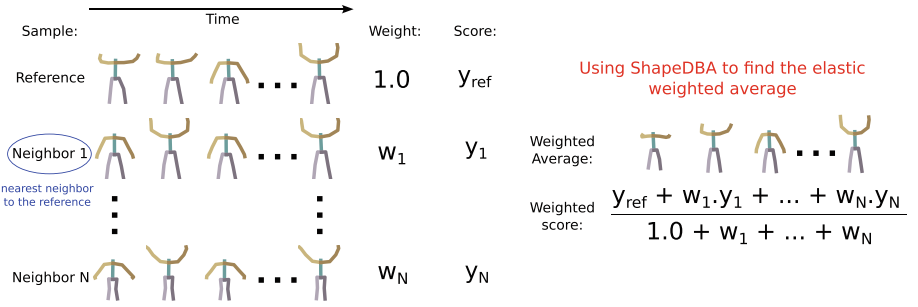


Fig. 1. Identifying N nearest neighbors for each reference in the dataset using Dynamic Time Warping (DTW). A weight is assigned to each neighbor based on the DTW measure to the reference. Using these weights, the weighted average sequence is computed via ShapeDBA, followed by calculating the weighted score to extend the regression training dataset with this new sample.

A common approach to address the lack of data in machine learning is data augmentation (DA), where new synthetic examples are created to increase the variability of the training data, improving the model's generalization capabilities. In the case of human rehabilitation assessment datasets, the most used method for generating new samples involves adding noise to existing real samples. While this technique can be effective in some cases [35], it may not always be the most appropriate approach, as it can lead to unrealistic and meaningless sequences.

Another way of generating synthetic sequences is using deep generative models, which have proven effective in other human motion tasks by generating realistic sequences [29]. However, these approaches present a paradox: if we do not have enough data to train a deep supervised model, how can we train a deep generative model on such a small dataset? To address this issue, researchers in the time series domain often rely on averaging techniques [28] aiming at creating average representative sequences from a set of training samples [18].

In this work, we propose to study the use of a recent averaging method, ShapeDBA [17], on skeleton-based rehabilitation sequences. This approach has been previously used to generate synthetic time series datasets through a weighted elastic averaging method called Dynamic Time Warping Barycenter Averaging (DBA). More recently, Ismail-Fawaz et al. [17] introduced a novel averaging method, ShapeDBA, which adapts DBA to more accurately reflect changes in sequence shapes over time. In this study, we utilize the weighted version of ShapeDBA approach to generate various synthetic average sequences. As such averaging method is time consuming, it cannot be computed at each epoch of the training phase, as usually done in data augmentation. Differently, we generate several average sequences to extend the size of the original training dataset, before model's training. We refer to this as data extension (DE) to differentiate it from data augmentation.

In this work, we target the task of rehabilitation assessment, which is often formulated as an extrinsic regression problem where the goal is to predict a continuous performance score associated to a rehabilitation sequence. In the state-of-the-art, most of the work have considered DA or DE for classification purposes, where each synthetic sample is associated with a discrete label corresponding to the original sample that has been augmented. However, associating a continuous label to synthetic data is not straightforward. We hence propose to leverage the weighting strategy by inferring a weighted continuous label computed from true labels of a given set of considered samples. Our proposed setup is detailed in Fig. 1.

It results in an extended dataset with both synthetic sequences and synthetic continuous labels, allowing deep learning models to generalize better for the task of rehabilitation assessment. The main contributions of this work are:

- We investigate the use of ShapeDBA for human skeleton sequences to generate average rehabilitation movements;
- We extend the averaging method by a weighting strategy that allows generating more various but yet coherent sequences;
- We leverage the weighting strategy for associating meaningful continuous labels to synthetic rehabilitation sequences;
- We consider the proposed method for extending a small real-world rehabilitation dataset and evaluate it on an extrinsic regression task.

2 Related Work

Skeleton-based approaches have shown promise for human motion analysis. Instead of considering the whole video stream, these approaches rely on

humanoid skeleton data extracted from each frame of the video, and allow capturing human posture and movement. This skeleton modality has also been explored for rehabilitation assessment in combination with classical Machine Learning algorithms such as Hidden Markov Models (HMM) [2] or Gaussian Mixture Models (GMM) [6]. More recently, Deep Learning approaches have been increasingly employed for rehabilitation assessment. One of the first Deep Learning methods applied to rehabilitation assessment from 3D skeleton data was proposed in [22], where a Long Short-Term Memory Network (LSTM) was considered. Inspired by the success of Convolutional Neural Network for time series classification [12,16,19], Ismayilzada et al. [20] investigated the use of various convolution-based architectures for rehabilitation assessment. Differently, a graph has been considered in [5] to represent the spatial and temporal relationship between human joints. Then, a Graph Convolutional Network (GCN) has been proposed for modeling and analysing the whole rehabilitation sequence. This promising method has been further extended in [24] by combining both positional and angular skeleton features with an attention fusion mechanism.

However, similarly to various fields, these Deep Learning models may suffer from overfitting and tend to not generalize well on new patients not seen during training [20]. This is particularly due to the lack of publicly available datasets for rehabilitation assessment. Moreover, while few datasets have been made available such as Kimore [3], UI-PRMD [32] or KERAAL [27], all of them include a limited number of rehabilitation sequences, thus complicating the learning task by deep models.

A well explored way of addressing data limitation and improving generalization of deep models is known as data augmentation (DA). DA involves introducing different variations in the training set at each epoch of the training phase, thus preventing the deep model to overfit the original training data. In 3D skeleton data, the most simple variations include geometric transformations [4] such as translation, rotation and scaling. However, such transformations do not affect the temporal dynamics of human motion, thus do not allow a deep model to learn various ways of performing rehabilitation exercises. To overcome this, many works consider applying random noise on skeleton sequences [4,31]. Differently, as human motion is considered as a spatio-temporal data, temporal DA [30] can also be considered, such as shifting [21] or warping [11]. While noisy or temporal augmentations have been shown to be effective for skeleton-based action recognition [35], they may result in incoherent motion as they do not consider the data distribution.

Conversely, generative approaches aim at creating synthetic data from a set of training samples. In recent years, many deep generative models have been proposed for human motion [14,29]. However, for rehabilitation assessment, such deep generative models may suffer from the limited amount of training data to correctly learn the data distribution and generate coherent rehabilitation sequences. In this case, other generative methods computing averages from a set of samples seem to be more appropriate. For temporal sequences, such as rehabilitation exercises, an elastic averaging approach is required to consider

temporal alignment. Hence, the DTW Barycenter Averaging (DBA) has been proposed in [28] by leveraging Dynamic Time Warping (DTW) [25] for temporal alignment. Similarly, a more recent averaging method called ShapeDBA [17] proposed to leverage the ShapeDTW measure [36], which is more robust to noise. However, such averaging methods are often computationally costly and thus cannot be computed at each epoch of the training phase. Instead, synthetic averages can be computed only once to increase the size of the original training set prior to the training phase. To differentiate this strategy from DA, we refer to it as data extension (DE). In this work, we consider skeleton DE and propose to use the recent ShapeDBA averaging method in a weighted version to generate meaningful synthetic rehabilitation sequences.

3 Proposed Approach

3.1 Skeleton Sequence Representation

In this work, we are interested in assessing physical rehabilitation exercises represented by 3D skeleton sequences extracted from RGB-D videos. For each frame t of a sequence, the 3D position of each joint j of the skeleton is represented by three coordinates $x_j(t)$, $y_j(t)$ and $z_j(t)$. Let N_j be the number of joints the skeleton is composed of, the skeleton pose $p(t)$ at frame t is then represented by a $3N_j$ dimensional vector:

$$p(t) = [x_1(t), y_1(t), z_1(t), \ldots, x_{N_j}(t), y_{N_j}(t), z_{N_j}(t)]. \tag{1}$$

The whole skeleton sequence S of length T is represented by a multivariate time series of shape $(3N_j, T)$:

$$S = [p(1), p(2), \ldots, p(T)]. \tag{2}$$

Such a multivariate time series S represents the evolution of a body pose p through time.

Each sequence is associated with a continuous label y representing a performance score, i.e., how much the rehabilitation exercise is correctly performed.

3.2 Sequence Averaging Using ShapeDBA

Sequence averaging is a task that is more commonly referred to as *prototyping* with the goal of finding a representative sample for a set of sequences. Given the nature of temporal ordering in sequences, especially human motion sequences, Petitjean et al. [28] argued the need of an elastic averaging approach to overcome the issues of traditional averaging techniques. The naive way of averaging temporal sequences is with the arithmetic mean; however, it assumes a perfect alignment between all samples, which is not the case in most datasets. Petitjean et al. [28] proposed the usage of elastic measures between sequences, such as Dynamic Time Warping (DTW) [25], and proposed DTW Barycenter Averaging (DBA). DTW is a similarity measure between two sequences that aligns two input sequences through a warping path, detecting some temporal distortions along the way, and performing the Euclidean Distance (ED) over the aligned series. DBA utilizes this algorithm through the following steps:

- **step 1:** Initialize the average sequence as one of the samples in the given dataset;
- **step 2:** Find the warping path between the average sequence and all other samples in the dataset;
- **step 3:** For each time stamp in the average sequence, replace its value with the barycenter of all aligned points from other samples in the dataset. The barycenter in this case refers to the arithmetic mean of all aligned values. Repeat **step 2** until convergence, meaning the average sequence no longer changes compared to a given threshold.

More recently, Ismail-Fawaz et al. in [17] argued that the usage of DTW in DBA can produce some out-of-distribution points in the output average sequence. This is due to the fact that aligning point-to-point between two time series presents some issues as argued in [36], for which they proposed ShapeDTW as a solution. ShapeDTW is a variant of DTW that aligns neighborhoods instead of points between two samples. Ismail-Fawaz et al. [17] hence proposed ShapeDBA, the DBA algorithm utilizing the ShapeDTW alignment method.

In this work, we utilize the ShapeDBA approach of averaging in order to create new samples and extend the training dataset.

3.3 Weighted Averages

As the manifold of rehabilitation motion sequences may be sparse, computing an average using the original shapeDBA method may result in a meaningless or incoherent average sequence. In order to follow the distribution of the motion sequences, a weighted average is more appropriate [7]. For computing a weighted average motion sequence, we employ the ShapeDBA algorithm and follow a similar strategy as in [7,15]. For a given reference motion sequence, we generate a synthetic version by considering a neighborhood of n motion sequences. A weight of 1 is associated to the reference sequence S_{ref}, while for neighboring sequences S_i, the weight depends on the similarity with the reference and is computed as:

$$w_i = e^{\ln(0.5) * \frac{DTW(S_i, S_{ref})}{d_{NN}}}, \tag{3}$$

where d_{NN} corresponds to the DTW distance between S_{ref} and its nearest neighbor. This allows to give more impact on nearest sequences, while computing the weighted average sequences. Figure 2 illustrates the importance of a weighted average when the considered manifold is sparse, and the distribution of motion sequences is not spherical.

For each reference sequence S_{ref}, the computation of the weighted ShapeDBA results in a corresponding synthetic sequence $\hat{S}$. In order to associate a continuous label (score) to the synthetic data, the same weights are used but are first normalized using a min-max normalization. This ensures the

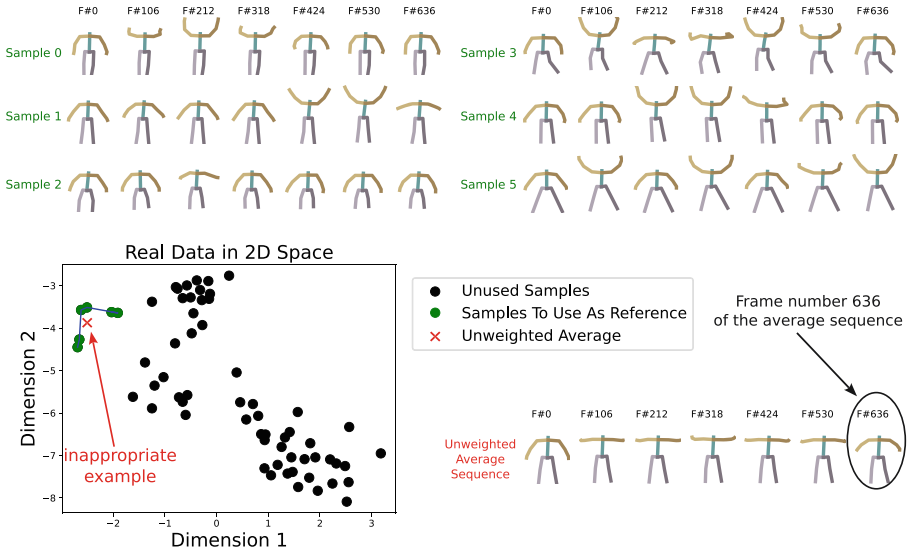


Fig. 2. Averaging six human motion sequences (from Example 0 to Example 5 on the top left/right) assuming a uniformly distributed weight produces an inappropriate example (bottom right skeleton sequence). Following the 2D t-SNE projection of these sequences (bottom left), the placement of the average sequence (bottom left in red) does not belong to the manifold of the original sequences (bottom left in green). (Color figure online)

weighted continuous labels keep in the original range between 0 and 1. Hence, the associated label $\hat{y}$ to a synthetic sequence $\hat{S}$ is computed as:

$$\hat{y} = \sum_{i=1}^{N+1} \bar{w_i} * y_i, \tag{4}$$

where $\bar{w_i}$ is the normalized weight associated to the i-th sequence in the considered set of size $N+1$ (the reference sequence and the N neighboring sequences).

4 Experimental Evaluation

We evaluated our proposed approach on two different aspects. The first one assesses the coherence of synthetic sequences, both qualitatively and quantitatively. The second investigates how synthetic sequences generated using our approach can be employed to help a deep learning model achieving extrinsic regression, i.e., predicting a continuous score associated with a rehabilitation sequence. The code is publicly available here:
https://github.com/MSD-IRIMAS/Weighted-ShapeDBA-4-Rehab

4.1 Experimental Setup

Dataset: We used the Kimore dataset [3] including RGB-D videos collected by a Kinect sensor from which 3D skeleton sequences have been extracted, for five distinct rehabilitation exercises. Each sequence is associated with an evaluation score provided by clinical experts, ranging from 0 to 100 (normalized to the range $[0, 1]$). In this work, we discarded rehabilitation sequences with missing skeleton values. It resulted in 71 sequences per exercise corresponding to 40 healthy subjects and 31 unhealthy patients. We resampled all the sequences to the average length of 748 frames using the Fourier method in the *scipy* [33]. We employed a 5-fold cross-validation protocol but differently from what is usually done in the literature, we slightly adapted the 5-fold split in order to consider unhealthy subjects at inference. We believed this is more close to a real-world scenario. As a result, only sequences corresponding to unhealthy subjects were split into 5 folds. For each run, all sequences corresponding to healthy subjects and 4 unhealthy folds were used for training, while the remaining unhealthy fold was used for test. We utilize *aeon* [23], a python toolkit for time series machine learning, for the similarity measures and ShapeDBA.

Comparative Sets of Rehabilitation Sequences: Our comparative study includes different sets of rehabilitation sequences. The *reference set*, of size D corresponds to original sequences from the Kimore dataset. As baseline, we created a noisy version of each reference sequence by adding a random noise $\mathcal{N}(0, 0.1)$ to all skeletons of the sequence. We referred to this set of size D as the *noisy set*. We then created 5 additional sets using the proposed method generating synthetic sequences. For each reference sequence from the *reference set*, we generated a synthetic version using weighted ShapeDBA, detailed in Sect. 3.3, by considering 5 different neighborhoods of size $N = 1$ to $N = 5$. It resulted in 5 new sets of size D, denoted as *ShapeDBA NN1*, *ShapeDBA NN2*, *ShapeDBA NN3*, *ShapeDBA NN4* and *ShapeDBA NN5*.

4.2 Evaluation of Synthetic Data Coherence

When it comes to generative models, it is mostly difficult to utilize one way of evaluation. For human motion data, one approach is to assess how realistic generated sequences are visually. However, it is not enough to use human visualisation as argued in [26], and metrics are needed to quantify how reliable generated samples are. In this section, we propose both ways to assess this reliability: visually and numerically.

Generation Metrics
The task of extending a dataset, even through non-deep learning methods, is still considered as a generation pipeline. For this reason, we found it crucial to assess the fidelity and diversity [26] of generated samples. Fidelity and diversity are both aspects of quantitative evaluation of generative models. On the one hand,

fidelity metrics evaluate how close the real and generated samples are. The closer they are the more *reliable* the generated samples are for real world applications in terms of fidelity. On the other hand, diversity metrics evaluate how far samples are from each other, independently for real and generated samples. Moreover, the goal of a generative model is as well, most of the times, to produce a generated space as diverse as the real one.

In this work, we relied on two common metrics to assess fidelity and diversity: The Fréchet Inception Distance (FID) and the Average Pair Distance (APD).

Fréchet Inception Distance (FID). The FID metric [10] assesses how close real and generated samples are in terms of distribution. For computing it, the real and generated samples first go through a latent feature extraction step using a pre-trained deep learning model $\mathcal{F}$ (in our case trained on a regression task). Second, the mean and covariance are calculated for both feature space. Third, FID computes the distance between both distributions assuming the mean and covariance are of a Gaussian distribution. The mathematical formulation of FID is defined below:

$$FID(\mathcal{P}_1, \mathcal{P}_2)^2 = trace(\Sigma_1 + \Sigma_2 - 2(\Sigma_1.\Sigma_2)^{1/2}) + \sum_{i=1}^{f}(\mu_{1,i} - \mu_{2,i})^2, \quad (5)$$

where $\mathcal{P}_1$ and $\mathcal{P}_2$ are the distributions of real and generated samples respectively, f is the dimension of the latent space of $\mathcal{F}$, μ_1 and μ_2 are both vectors of dimension f representing the mean over all samples in the feature space of real and generated samples respectively. Σ_1 and Σ_2 are both matrices of dimension (f, f) representing the covariance matrices over all samples in the feature space of real and generated samples, respectively.

Average Pair Distance (APD). The APD metric [9] is calculated independently on each of the real and generated samples. The metric calculates the average ED between randomly selected samples in the feature space, using $\mathcal{F}$ as the latent feature extractor. The APD defines two randomly selected sets of samples, $\mathcal{S}_1$ and $\mathcal{S}_2$, of S_{apd} samples each and produces the average distance between these two sets as follows:

$$APD(\mathcal{S}, \mathcal{S}^{'}) = \frac{1}{S_{apd}} \sum_{i=1}^{S_{apd}} \sqrt{\sum_{j=1}^{f}(\mathcal{S}_{i,j} - \mathcal{S}^{'}_{i,j})^2}, \quad (6)$$

where the above is calculated on many different sets of $\mathcal{S}_1$ and $\mathcal{S}_2$ and the final presented value is the average APD found over different experiments. This is necessary to remove any bias in the selection of the two sets.

In this work, we utilized the open source software made available by [13] to calculate the above two metrics. The presented results are averaged over different initialization of the pre-trained regression model, which in our case is the deep regression model, trained only on the training set. We used a value of $S_{apd} = 20$ for the size of the randomly selected sets when calculating the APD metric.

Qualitative Analysis: Visualizing Real vs Generated. In Fig. 3, we present three sequences, the first being a real sequence from the Kimore dataset, the second being generated by simply adding noise to the real sequence. The third sequence is generated using our weighted ShapeDBA approach detailed in Sect. 3.3. It can be seen that, visually, the noisy sample is not realistic as much as the weighted ShapeDBA generation.

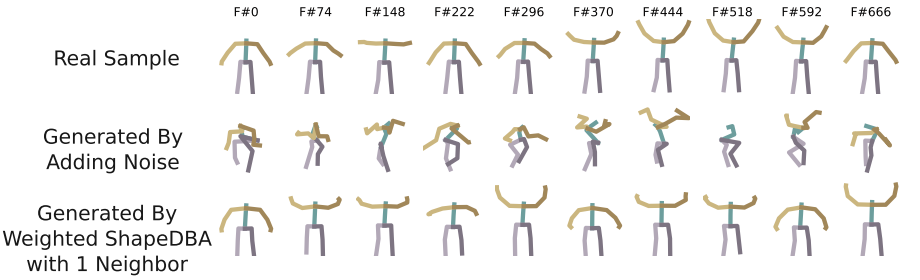


Fig. 3. Visualization of three examples: real sample (top), noisy sample (middle) and generated sample through weighted ShapeDBA (bottom). For each of the three sequences, we present 10 frames, counting from left to right.

To visualize the impact of weighted ShapeDBA, we present in Fig. 4, two sequences and the produced weighted ShapeDBA following the formula in Sect. 3.3. It can be seen from the figure, that given the higher weight is associated to the first sequence (the reference), the motion over the temporal axis is very similar and aligned exactly with the reference. However, given a weight is assigned to the second sequence, the produced weighted ShapeDBA contains some information from the second sequence such as the height and form of the patient from whom the sequence is recorded.

Quantitative Analysis: Fidelity and Diversity. Visualizing generated sequences may produce a good enough satisfaction, however there should be a quantitative evaluation as argued in [26]. We relied on the two metrics presented in Sect. 4.2, the FID for fidelity evaluation and the APD for diversity evaluation. For each augmentation method, i.e., noisy and weighted ShapeDBA, we present the average FID and APD values, as well as the standard deviation. These values are aggregated over different resamples of the dataset for each exercise, as well as different initialization of the pre-trained deep regression model. In Table 1, we present the values of the FID for each exercise using the noisy augmentation method and five versions of our weighted ShapeDBA (neighbors range from 1 to 5). We also include the FID value over the real samples, which serves as a baseline, as argued in [13]. The FID of any generation method should have a close but still higher value to the computed FID on real data. This is due to the fact that generated samples cannot be more reliable, in terms of fidelity, than the real set of data with itself. It can be seen from Table 1 that the FID of

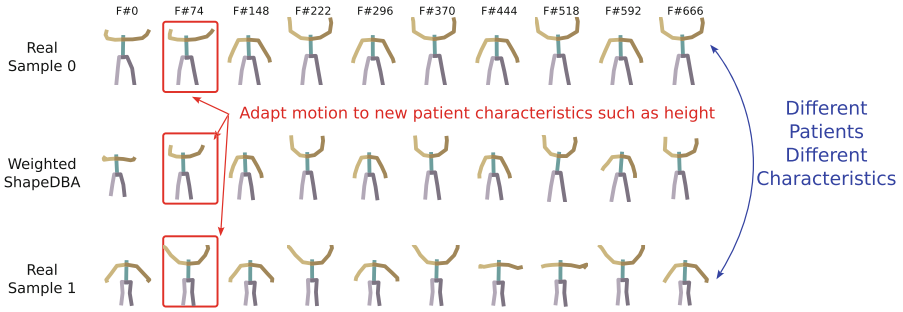


Fig. 4. Visualizing two real sequences (top and bottom) and their produced weighted ShapeDBA sequence (middle). The produced average sequence kept the temporal alignment of the exercise from the top sequence as it is associated with a higher weight than the bottom one. However the characteristic of the patient (height) is kept in the average middle sequence.

the noisy method is smaller than the one of the weighted ShapeDBA method. Discarding the visualization done in previous qualitative evaluation, the noisy augmentation wins over the quantitative evaluation. However from Fig. 3, it can be obviously seen that the noisy augmentation method produces meaningless sequences. This is argued in [13], as there is no one *best* way to evaluate generative models, instead many approaches should be used to reach any concrete conclusion.

The same interpretation is argued for the APD values in Table 2, as the noisy method comes out as winner instead of the weighted ShapeDBA method as it has the closest value to the APD on real samples.

Table 1. The FID values of different augmentation methods and the real dataset, over different resamples of each exercise. The presented FID values include the average and standard deviation over all resamples per exercise and different initialization of the pre-trained feature extractor, **best value** second best value.

	Exercise 1	Exercise 2	Exercise 3	Exercise 4	Exercise 5
Real	$02.18E^{-6} \pm 06.66E^{-7}$	$02.48E^{-6} \pm 09.82E^{-7}$	$02.19E^{-6} \pm 08.18E^{-7}$	$05.00E^{-6} \pm 02.09E^{-6}$	$03.06E^{-6} \pm 01.30E^{-6}$
Noisy	**00.07 ± 00.03**	**00.08 ± 00.03**	**00.09 ± 00.03**	**00.24 ± 00.10**	**00.07 ± 00.03**
ShapeDBA NN1	01.94 ± 00.81	04.12 ± 01.86	02.28 ± 01.10	04.19 ± 02.95	03.39 ± 01.92
ShapeDBA NN2	03.15 ± 01.06	06.62 ± 02.56	04.01 ± 02.70	05.95 ± 04.07	05.75 ± 02.34
ShapeDBA NN3	03.77 ± 01.21	07.98 ± 02.59	05.16 ± 03.39	07.16 ± 04.51	07.34 ± 02.91
ShapeDBA NN4	04.31 ± 01.24	09.38 ± 03.05	05.96 ± 03.52	08.01 ± 04.64	08.51 ± 03.76
ShapeDBA NN5	04.62 ± 01.28	10.21 ± 03.34	06.45 ± 03.69	08.90 ± 05.07	09.23 ± 04.12

4.3 Evaluation of Synthetic Sequences for Data Extension

Our second experiment evaluates the proposed approach as data extension for rehabilitation assessment. The rehabilitation assessment task is an extrinsic regression problem, where the goal is to predict a continuous value associated to a rehabilitation sequence, i.e., the performance score.

Table 2. The APD values of different augmentation method and the real dataset, over different resamples of each exercise. The presented APD values include the average and standard deviation over all resamples per exercise, different randomly selected sets of size $S_{apd} = 20$ and different initialization of the pre-trained feature extractor.

	Exercise 1	Exercise 2	Exercise 3	Exercise 4	Exercise 5
Real	06.09 ± 00.63	06.61 ± 00.83	05.95 ± 00.78	08.24 ± 01.24	07.10 ± 01.10
Noisy	**06.05 ± 00.57**	**06.55 ± 00.79**	**05.90 ± 00.78**	**08.14 ± 01.18**	**07.00 ± 01.03**
ShapeDBA NN1	05.21 ± 00.68	05.32 ± 00.63	05.06 ± 00.68	07.14 ± 01.12	06.11 ± 01.08
ShapeDBA NN2	04.93 ± 00.69	04.96 ± 00.66	04.77 ± 00.65	06.90 ± 01.16	05.70 ± 00.98
ShapeDBA NN3	04.78 ± 00.75	04.64 ± 00.54	04.53 ± 00.66	06.84 ± 01.20	05.52 ± 00.96
ShapeDBA NN4	04.67 ± 00.74	04.47 ± 00.55	04.34 ± 00.63	06.59 ± 01.16	05.35 ± 00.91
ShapeDBA NN5	04.61 ± 00.67	04.35 ± 00.53	04.30 ± 00.61	06.50 ± 01.13	05.31 ± 00.90

Fully Convolutional Network for Extrinsic Regression: For the task of rehabilitation assessment, we employed a Fully Convolutional Network [34]. We used the same architecture and same hyperparameters as in [19], except for the last layer that includes a single neuron with sigmoid activation for predicting a single continuous score. A detailed view of the FCN architecture used in this work for the regression task is presented in Fig. 5. The FCN architecture consists of three stacked convolution blocks followed by an aggregation layer, Global Average Pooling, before being fed to a Fully Connected layer to predict one single value. Each convolution block consists on a one dimensional convolution layer, a Batch Normalization layer in order to re-scale the features to the same range of values, and a ReLU activation in order to detect only the activated features of each convolution filter.

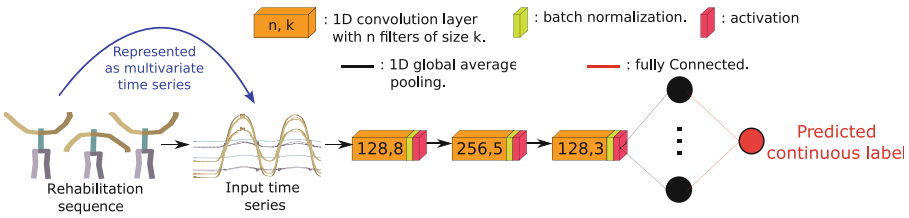


Fig. 5. The Fully Convolutional Network (FCN) used for the task of extrinsic regression on rehabilitation sequences represented as multivariate time series.

Regression Metrics: For comparing the score predicted by the considered models with the clinical score provided by experts, we followed [2] and used two metrics: the Root Mean Square Error (RMSE), and the Mean Absolute Error (MAE). Given two sets of N scores, $\mathbf{y}$ as the ground truth and $\hat{\mathbf{y}}$ as the predictions, both the MAE and RMSE are computed as follows:

$$MAE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|, \tag{7}$$

$$RMSE(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}. \tag{8}$$

Quantitative Analysis: We evaluated the performance of the considered FCN model for extrinsic regression, while trained on the different sets of rehabilitation sequences described in Sect. 4.1, concatenated with the *reference set*. Average MAE and RMSE errors (± std) are reported in Table 3. We can first notice that, in most of the cases, a better performance is obtained when noisy sequences are added to the training set (*Ref. + Noise*), with respect to the case where only the original training set is used (*Ref.*). This shows the FCN model overfits the training set and considering data extension allows slightly improving generalization.

Moreover, it can be seen from Table 3 that the best error values for both metrics are obtained when the FCN model is trained on the rehabilitation set extended with ShapeDBA averages. This shows that the synthetic average sequences not only allow preventing overfitting but also correspond to more realistic rehabilitation motion so that a FCN model can better learn variations of rehabilitation exercises.

Finally, we can observe that, for three exercises among five, the best performance is obtained when training data includes average sequences computed using a single neighbor (*Ref. + ShapeDBA NN1*). This suggests that considering a larger neighborhood may result in incoherent average sequences in some cases as neighbors are not lying in a continuous subspace.

Table 3. MAE and RMSE errors obtained for all compared approaches on each exercise separately. Best values are emphasized in bold, while second best values are underlined.

Training Set	Exercise 1	Exercise 2	Exercise 3	Exercise 4	Exercise 5
MAE					
Ref.	0.206 ± 0.069	0.202 ± 0.037	0.204 ± 0.055	0.184 ± 0.068	0.224 ± 0.058
Ref. + Noise	0.186 ± 0.065	**0.172 ± 0.040**	0.203 ± 0.045	0.185 ± 0.073	0.229 ± 0.069
Ref. + ShapeDBA NN1	0.167 ± 0.070	0.175 ± 0.030	**0.182 ± 0.051**	**0.141 ± 0.062**	**0.208 ± 0.079**
Ref. + ShapeDBA NN2	0.169 ± 0.057	0.177 ± 0.041	0.194 ± 0.041	0.168 ± 0.056	0.226 ± 0.066
Ref. + ShapeDBA NN3	0.173 ± 0.063	0.183 ± 0.047	0.199 ± 0.058	0.168 ± 0.083	0.225 ± 0.055
Ref. + ShapeDBA NN4	0.168 ± 0.059	0.179 ± 0.043	0.199 ± 0.043	0.180 ± 0.080	0.231 ± 0.060
Ref. + ShapeDBA NN5	**0.166 ± 0.067**	0.185 ± 0.043	0.201 ± 0.050	0.182 ± 0.089	0.226 ± 0.061
RMSE					
Ref.	0.251 ± 0.083	0.247 ± 0.045	0.248 ± 0.065	0.230 ± 0.083	0.267 ± 0.073
Ref. + Noise	0.203 ± 0.078	0.226 ± 0.043	0.238 ± 0.046	0.227 ± 0.090	0.274 ± 0.092
Ref. + ShapeDBA NN1	0.199 ± 0.087	**0.226 ± 0.036**	**0.214 ± 0.054**	**0.178 ± 0.074**	**0.251 ± 0.094**
Ref. + ShapeDBA NN2	0.203 ± 0.075	0.232 ± 0.052	0.226 ± 0.044	0.210 ± 0.074	0.268 ± 0.083
Ref. + ShapeDBA NN3	0.205 ± 0.082	0.235 ± 0.050	0.240 ± 0.062	0.214 ± 0.105	0.268 ± 0.066
Ref. + ShapeDBA NN4	**0.198 ± 0.071**	0.235 ± 0.050	0.234 ± 0.048	0.230 ± 0.105	0.279 ± 0.070
Ref. + ShapeDBA NN5	0.202 ± 0.079	0.230 ± 0.049	0.244 ± 0.057	0.231 ± 0.109	0.280 ± 0.080

5 Conclusion

In this work, we addressed the problem of rehabilitation assessment with limited data by employing a state-of-the-art elastic averaging method, ShapeDBA, to extend the amount of available training data. Our approach leverages weighted averaging to generate synthetic rehabilitation sequences and associate new continuous scores with them, thus enhancing the assessment process. Experimental evaluation demonstrated the promise of this method to generate coherent rehabilitation sequences.

However, our method remains dependent on the real data distribution, particularly for continuous labels, which limits control over the generated sequences and scores. Future work will focus on evaluating this approach on more diverse datasets and models as well as investigating how to improve control over synthetic sequences, ultimately aiming to enhance data-driven rehabilitation assessment and patient outcomes.

Acknowledgment. This work was supported by the ANR DELEGATION project (grant ANR-21-CE23-0014) of the French Agence Nationale de la Recherche. The authors would like to acknowledge the High Performance Computing Center of the University of Strasbourg for supporting this work by providing scientific support and access to computing resources. Part of the computing resources were funded by the Equipex Equip@Meso project (Programme Investissements d'Avenir) and the CPER Alsacalcul/Big Data. The authors would also like to thank the creators and providers of the Kimore dataset.

References

1. Blanchard, A., Nguyen, S.M., Devanne, M., Simonnet, M., Le Goff-Pronost, M., Rémy-Néris, O.: Technical feasibility of supervision of stretching exercises by a humanoid robot coach for chronic low back pain: the r-cool randomized trial. BioMed Res. Int. **2022**(1), 1–10 (2022). https://doi.org/10.1155/2022/5667223
2. Capecci, M., et al.: A hidden semi-Markov model based approach for rehabilitation exercise assessment. J. Biomed. Inform. **78**, 1–11 (2018)
3. Capecci, M., et al.: The kimore dataset: kinematic assessment of movement and clinical scores for remote monitoring of physical rehabilitation. IEEE Trans. Neural Syst. Rehabil. Eng. **27**(7), 1436–1448 (2019)
4. Chen, J., Yang, W., Liu, C., Yao, L.: A data augmentation method for skeleton-based action recognition with relative features. Appl. Sci. **11**(23), 11481 (2021)
5. Deb, S., Islam, M.F., Rahman, S., Rahman, S.: Graph convolutional networks for assessment of physical rehabilitation exercises. IEEE Trans. Neural Syst. Rehabil. Eng. **30**, 410–419 (2022)
6. Devanne, M., et al.: Multi-level motion analysis for physical exercises assessment in kinaesthetic rehabilitation. In: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pp. 529–534. IEEE (2017)
7. Forestier, G., Petitjean, F., Webb, G., Dau, H.A., Keogh, E.: Generating synthetic time series to augment sparse datasets. In: IEEE International Conference on Data Mining (ICDM), pp. 865–870 (2017). https://doi.org/10.1109/ICDM.2017.106

8. Guijo-Rubio, D., Middlehurst, M., Arcencio, G., Silva, D.F., Bagnall, A.: Unsupervised feature based algorithms for time series extrinsic regression. Data Min. Knowl. Discov. 1–45 (2024)
9. Guo, C., et al.: Action2motion: conditioned generation of 3d human motions. In: Proceedings of the 28th ACM International Conference on Multimedia, pp. 2021–2029 (2020)
10. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
11. Huynh-The, T., Hua, C.H., Kim, D.S.: Encoding pose features to images with data augmentation for 3-D action recognition. IEEE Trans. Industr. Inf. **16**(5), 3100–3111 (2019)
12. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Lite: light inception with boosting techniques for time series classification. In: 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10. IEEE (2023)
13. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Establishing a unified evaluation framework for human motion generation: a comparative analysis of metrics. arXiv preprint arXiv:2405.07680 (2024)
14. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: A supervised variational auto-encoder for human motion generation using convolutional neural networks. In: 4th International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI) (2024)
15. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Weighted elastic barycetner averaging to augment time series data (2024). https://github.com/MSD-IRIMAS/Augmenting-TSC-Elastic-Averaging
16. Ismail-Fawaz, A., Devanne, M., Weber, J., Forestier, G.: Deep learning for time series classification using new hand-crafted convolution filters. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 972–981. IEEE (2022)
17. Ismail-Fawaz, A., et al.: Shapedba: generating effective time series prototypes using shapedtw barycenter averaging. In: International Workshop on Advanced Analytics and Learning on Temporal Data, pp. 127–142. Springer (2023)
18. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Data augmentation using synthetic data for time series classification with deep residual networks. In: ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (2018)
19. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Min. Knowl. Disc. **33**(4), 917–963 (2019)
20. Ismayilzada, E., Devanne, M., , Weber, J., Forestier, G.: Time series extrinsic regression for physical rehabilitation assessment. In: Upper Rhine Artificial Intelligence Symposium (URAI) (2023). undefined
21. Lee, I., Kim, D., Kang, S., Lee, S.: Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1012–1020 (2017)
22. Liao, Y., Vakanski, A., Xian, M.: A deep learning framework for assessing physical rehabilitation exercises. IEEE Trans. Neural Syst. Rehabil. Eng. **28**(2), 468–477 (2020)
23. Middlehurst, M., et al.: aeon: a python toolkit for learning from time series. arXiv preprint arXiv:2406.14231 (2024)

24. Mourchid, Y., Slama, R.: MR-STGN: Multi-residual Spatio temporal graph network using attention fusion for patient action assessment. In: 2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP), pp. 1–6. IEEE (2023)
25. Müller, M.: Dynamic time warping. Inf. Retrieval Music Motion, 69–84 (2007)
26. Naeem, M.F., Oh, S.J., Uh, Y., Choi, Y., Yoo, J.: Reliable fidelity and diversity metrics for generative models. In: International Conference on Machine Learning, pp. 7176–7185. PMLR (2020)
27. Nguyen, S., Devanne, M., Remy Neris, O., Lempereur, M., Thepaut, A.: A medical low-back pain physical rehabilitation database for human body movement analysis. In: International Joint Conference on Neural Networks (IJCNN). IEEE (2024)
28. Petitjean, F., Ketterlin, A., Gançarski, P.: A global averaging method for dynamic time warping, with applications to clustering. Pattern Recogn. **44**(3), 678–693 (2011)
29. Petrovich, M., Black, M.J., Varol, G.: Action-conditioned 3d human motion synthesis with transformer VAE. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10985–10995 (2021)
30. Pialla, G., Devanne, M., Weber, J., Idoumghar, L., Forestier, G.: Data augmentation for time series classification with deep learning models. In: International Workshop on Advanced Analytics and Learning on Temporal Data (2022)
31. Rao, H., Xu, S., Hu, X., Cheng, J., Hu, B.: Augmented skeleton based contrastive action learning with momentum LSTM for unsupervised action recognition. Inf. Sci. **569**, 90–109 (2021)
32. Vakanski, A., Jun, H.P., Paul, D., Baker, R.: A data set of human body movements for physical rehabilitation exercises. Data **3**(1), 2 (2018)
33. Virtanen, P., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods **17**(3), 261–272 (2020)
34. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: a strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1578–1585. IEEE (2017)
35. Xin, C., Kim, S., Cho, Y., Park, K.S.: Enhancing human action recognition with 3D skeleton data: a comprehensive study of deep learning and data augmentation. Electronics **13**(4), 747 (2024)
36. Zhao, J., Itti, L.: shapeDTW: shape dynamic time warping. Pattern Recogn. **74**, 171–184 (2018)

Author Index

V. Lemaire et al. (Eds.): AALTD 2024, LNAI 15433, p. 147, 2025.
https://doi.org/10.1007/978-3-031-77066-1