Convolutional- and Deep Learning-Based Techniques for Time Series Ordinal Classification

Rafael Ayllón-Gavilán, David Guijo-Rubio, *Member, IEEE*, Pedro Antonio Gutiérrez, *Senior Member, IEEE*, Anthony Bagnall, and César Hervás-Martínez, *Senior Member, IEEE*

***Abstract*—Time-series classification (TSC) covers the supervised learning problem where input data is provided in the form of series of values observed through repeated measurements over time, and whose objective is to predict the category to which they belong. When the class values are ordinal, classifiers that take this into account can perform better than nominal classifiers. Time-series ordinal classification (TSOC) is the field bridging this gap, yet unexplored in the literature. There are a wide range of time-series problems showing an ordered label structure, and TSC techniques that ignore the order relationship discard useful information. Hence, this article presents the first benchmarking of TSOC methodologies, exploiting the ordering of the target labels to boost the performance of current TSC state of the art. Both convolutional- and deep-learning-based methodologies (among the best performing alternatives for nominal TSC) are adapted for TSOC. For the experiments, a selection of 29 ordinal problems has been made. In this way, this article contributes to the establishment of the state of the art in TSOC. The results obtained by ordinal versions are found to be significantly better than current nominal TSC techniques in terms of ordinal performance metrics, outlining the importance of considering the ordering of the labels when dealing with this kind of problems.**

***Index Terms*—Ordinal classification, time-series analysis, time-series classification (TSC), time-series machine learning (ML).**

Received 13 July 2024; revised 28 October 2024 and 5 November 2024; accepted 12 November 2024. Date of publication 27 November 2024; date of current version 3 February 2025. This work was supported in part by the "Agencia Estatal de Investigación (España)" under Grant PID2023-150663NB-C22/AEI/10.13039/501100011033; in part by the U.K. Engineering and Physical Sciences Research Council under Grant EP/W030756/2; in part by the European Commission, AgriFoodTEF under Grant DIGITAL-2022-CLOUD-AI-02 and Grant 101100622; in part by the Secretary of State for Digitalization and Artificial Intelligence ENIA International Chair under Grant TSI-100921-2023-3; and in part by the University of Córdoba and Junta de Andalucía under Grant PP2F_L1_15. The work of Rafael Ayllón-Gavilán was supported in part by the "Instituto de Salud Carlos III" (ISCIII), and in part by EU under Grant FI23/00163. This article was recommended by Associate Editor C.-T. Lin. *(Corresponding author: David Guijo-Rubio.)*

Rafael Ayllón-Gavilán is with the Department of Clinical-Epidemiological Research in Primary Care, Instituto Maimónides de Investigación Biomédica de Córdoba, 14004 Córdoba, Spain (e-mail: rafael.ayllon@imibic.org).

David Guijo-Rubio, Pedro Antonio Gutiérrez, and César Hervás-Martínez are with the Department of Computer Science and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain (e-mail: dguijo@uco.es; pagutierrez@uco.es; chervas@uco.es).

Anthony Bagnall is with the Department of Electronics and Computer Science, University of Southampton, SO17 1BJ Southampton, U.K. (e-mail: A.J.Bagnall@soton.ac.uk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2024.3498100.

Digital Object Identifier 10.1109/TCYB.2024.3498100

I. Introduction

A TIME series is an ordered sequence of values. This type of data is found in a wide variety of domains, such as medicine [1], financial analysis [2], and agriculture [3]. For instance, electrocardiogram signals [4] are ordered by time, and spectrograms [5] are ordered by frequency. There are numerous tasks applicable to time series, such as forecasting the next value [6], [7], clustering time series into groups without class information [8], [9], performing extrinsic regression on time series [10], and detecting anomalies within the data [11], among others. However, time-series classification (TSC) is the most popular machine learning (ML) tasks, with hundreds of different approaches proposed in [12] and [13]. For instance, [14] proposed a fuzzy-driven methodology, while [15] focused on classifying time series based on the distance between them. More recent approaches aim to classify time series using subsequences that capture the characteristics of the entire series [16], exploring the use of transformers applied to time series [17], and reservoir models based on spiking neural P systems [18]. Moreover, other time-series-related domains are time-series segmentation [19] and the discovery of motifs [20], [21], with an increasing interest in the last years.

TSC involves predicting a discrete output variable for a given time series. Depending on the number of variables observed at each time point, time series are univariate (only one channel) or multivariate (two or more channels) [22], [23]. The publication of the TSC archive[1] [24] allowed the development of effective methods for TSC. This archive provides a heterogeneous problem set that facilitates objective comparisons of new algorithms: a recent bake off study compared 33 TSC algorithms proposed in the last five years using the UCR archive [13].

While there has been significant progress in the algorithmic development for TSC, almost no attention has been paid to time-series ordinal classification (TSOC). We aim to address this absence. Some problems in the UCR archive are ordinal in nature, i.e., labels associated with samples follow an ordinal relationship. Up to now, these problems have been tackled by nominal TSC methods, which can limit the learning process: nominal methods generally require more data or iterations to achieve the same performance as an ordinal classifier [25].

We can define ordinal classification (also known as ordinal regression) as a classification problem where the output labels

[1]https://timeseriesclassification.com/

exhibit a natural ordering. This characteristic can be found in many and varied domains, such as human age prediction [26], climatological applications [27], and medical research [28]. This research takes advantage of information related to the ordering of the categorical labels to increase the performance of the models being applied. The most prominent type of models in the literature on ordinal classification are the so-called threshold-based models. In these methodologies the existence of a real-valued variable underlying the ordinal response is assumed. Hence, the training process focuses on modeling the real variable and learning the optimal thresholds determining which intervals corresponds to each ordinal label. The cumulative link model (CLM) approach belongs to this family of threshold-based models, and works with cumulative probabilities of the input belonging to a certain class or classes lower in the ordinal scale.

An example of ordinal classification can be seen in [28], where a set of patients infected with the SARS-CoV-2 coronavirus is studied and grouped into three levels of illness severity: 1) moderate; 2) severe; and 3) critical. These labels follow an ordinal relationship, in that a critical patient is sicker than a moderate or a severe patient. The main characteristic of an ordinal variable, using this example, is that it is worse to misclassify a critical patient as moderate, than it is to misclassify them as severe. The magnitude of the error should be higher in the first case.

As stated in [29], two types of ordinal problems can be distinguished in the literature: 1) grouped continuous variables and 2) assessed ordered categorical variables. The former involves an underlying continuous variable that is divided into different categories through a discretization procedure. In contrast, the latter does not involve a continuous variable, and instead, a domain expert assigns labels to patterns, establishing the ordering based on their judgment. An example of the first type is predicting the price of a computer for the next week in Euros in several categories (e.g., 0–1000, 1001–2000, 2001–3000, and so on). In this case, the categories are more objective as they are not influenced by human opinion. On the other hand, an example of the second type is assessing the severity of an investment risk based on its trajectory (e.g., none, mild, moderate, severe, and critical). In this scenario, the categories may be subjective, as different experts in the domain may have varying opinions and could assign the same risk to different trajectories. In this work, our focus lies on the first type of ordinal problems, as all the TSOC problems sourced from various data repositories involve the discretization of an underlying continuous variable. However, the methods are also applicable to the second type of problems.

An example of dataset used in this study is the DistalPhalanxOAG dataset [30]. It is specifically designed to assess the effectiveness of detecting hand and bone outlines and determine if they can aid in predicting bone age. The dataset focuses on distal phalanges of the middle finger, and the labels correspond to different age groups: 0–6 years old, 7–12 years old, and 13–19 years old. Fig. 1 illustrates three patterns from each of the classes, showcasing the observable ordinality of the labels in certain parts of the time series.

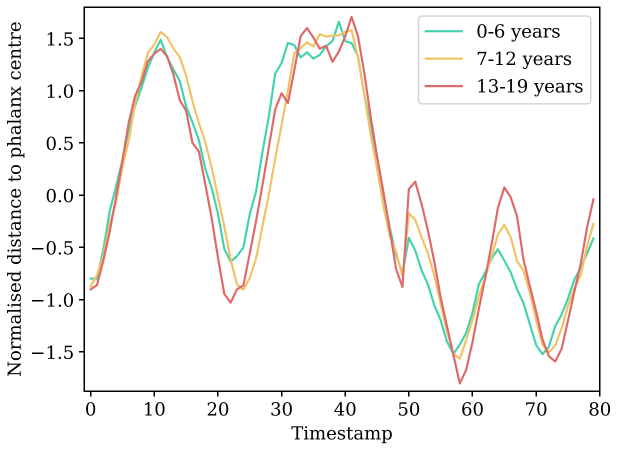


Fig. 1. Example of time-series extracted from the *DistalPhalanxOAG* dataset. The target ordinal scale represents different age ranges.

Other domains that could benefit from the development of the TSOC field include cardiology [4], where the objective is to develop methodologies for predicting the spontaneous termination of atrial fibrillations (AFs). Here, the input time series consists of two-channel electrocardiogram recordings from patients affected by AFs, and the aim is to predict whether the AF will terminate within 1 s, 1 h, or more than 1 h. In the field of spirit authentication, researchers in [5] utilized noninvasive near infrared spectroscopy time series to classify alcohol content into one of the following levels: E35, E38, E40, and E45, which clearly demonstrate an order relationship. Furthermore, another area is in stock market prediction [31]. The authors presented a methodology for predicting the direction (decreasing, stationary, and increasing) of the mid-price for various prediction horizons. In this scenario, there exists a natural order between the labels, making it suitable for TSOC analysis.

Our contributions can be listed as follows.

1) The development of seven novel TSOC techniques, focusing on the state-of-the-art approaches in TSC.
2) The identification of 29 ordinal time-series problems from different data sources (TSC, time-series extrinsic regression (TSER), and development of new TSOC datasets), and introducing the University of Córdoba (UCO) TSOC repository.
3) A benchmarking study of all these seven techniques over the whole set of 29 ordinal datasets.
4) The proposed approaches have been compared against their nominal counterpart (TSC approaches), the state-of-the-art HIVE-COTE2 (HC2), and standard ML approaches. The results demonstrate that TSOC techniques significantly enhance existing state-of-the-art nominal TSC methods in ordinal problems. Notably, HC2 is overall outperformed by TSOC methodologies, particularly by O-MiniROCKET, including in terms of accuracy.
5) Contributing to the literature with the first baseline study for TSOC, which aims to encourage time-series community for further enhancement in this novel field.

The remainder of this article is organized as follows. Related works are described in Section II. Section III formalizes preliminary definitions needed to present the different proposals. Section IV defines the proposed TSOC methods. Section V specifies the experimental setup and the datasets considered. Finally, in Section VI, we provide the conclusions and future research of our work.

II. Related Works

A. Time-Series Classification

A range of TSC methodologies have been developed in recent years. The first taxonomy grouping these approaches by their typology was proposed in [12], and extended in [13]. This taxonomy categorizes methodologies into eight distinct families: 1) distance-based; 2) interval-based; 3) shapelets-based; 4) feature-based; 5) dictionary-based; 6) convolutional-based; 7) deep-learning (DL)-based; and 8) hybrid methods. In this work, we focus on two of the best performing categories: convolutional-based and DL-based techniques. In the following, we provide a comprehensive introduction to both.

Convolutional-Based Techniques: This family of methods tries to extract features from the input time series by applying a set of convolutional kernels. The random convolutional kernel transform (ROCKET) method [32] was the first approach in this group of algorithms. It consists in training a Ridge regressor with features extracted from the application of kernels to the time series. In addition, two enhanced versions of this method were presented to the literature: 1) MiniROCKET [33], which proposes a computationally lighter kernel extraction process while maintaining a high accuracy and 2) MultiROCKET [34], which extends the kernel extraction by applying it to the first-order difference of the time series and performing more pooling operators. MultiROCKET is significantly better than MiniROCKET in accuracy, but it is computationally more expensive. A recently established method in the literature is the hybrid dictionary-rocket architecture (HYDRA) [35]. HYDRA is a combination of dictionary and convolutional techniques that achieves a highly competitive performance in comparison with existing approaches. HYDRA can be used in combination with any ROCKET-based methodology. In the current state of the art, MultiROCKET combined with HYDRA (HYDRA+MultiROCKET) stands as the best TSC methodology when accounting for both accuracy and training time. Additionally, several recent approaches utilize ROCKET as the foundational method. Uribarri et al. [36] proposed Detach-ROCKET, which introduces a sequential feature detachment step in the pipeline to identify and prune nonessential features from the feature extraction phase in ROCKET-based models. Foumani et al. [17] combined two position encodings within a transformer block alongside a convolutional layer, resulting in a novel convolutional approach that achieves competitive results compared to other state-of-the-art TSC methods.

DL-Based Models: DL has also been applied to TSC. Among the most commonly used architectures for neural networks are the multilayer perceptron (MLP) [37] and the convolutional neural network (CNN), which typically serve as a baseline architecture for DL-based TSC methods. Notably, InceptionTime [38] stands out as the leading methodology within this category, being an ensemble of five Inception Networks, which, as with other approaches, such as ResNet [37], is based on residual networks. Various adaptations and extensions of InceptionTime have been explored in the literature. For instance, InceptionFCN combines the fully connected network (FCN) with the InceptionTime architecture [10], while H-InceptionTime employs handcrafted filters in the convolutional layers of the network [39]. Recently, LITE has been introduced as a novel architecture featuring a light inception module (IM) with boosting techniques, using less than 3% of InceptionTime's number of parameters [40]. Recurrent neural networks (RNNs) is another popular type of architectures for DL. However, their application in TSC is relatively limited, primarily due to their design, which focuses on predicting an output for each timestamp in the time series [41]. Long short-term memory (LSTM)-based approaches have also been proposed for addressing the TSC task. For example, [42] introduced a difference-guided representation learning network that utilized LSTM to model the temporal dependencies and dynamic evolution of time-series data. Additionally, transformer-based DL has gained in popularity for TSC, often employing a straightforward encoder structure comprising attention and feedforward layers. Notable among these approaches is AutoTransformer [43], which uses a neural architecture search algorithm to identify the most suitable network architecture before passing the output to a multiheaded attention block. Furthermore, graph neural networks [44], [45] have recently been developed, demonstrating their potential due to their ability to model intertemporal/channel relationships, areas where other DL-based methods often face challenges [7].

Despite the progress and strong performance achieved with existing methodologies in TSC, the performance achieved for the ordinal problems could be significantly improved with appropriate approaches. Following, we specifically focus on ordinal classifiers.

B. Ordinal Classification

Over the last few years, several ordinal classification techniques based on nominal methodologies have been developed to exploit ordinality in the training process. In [46], a novel learning algorithm based on large margin rank boundaries was proposed for ordinal classification tasks, and the result can be seen as an adaptation of the support vector machine (SVM) algorithm to ordinal problems. Subsequently, methods, such as support vector ordinal regression with explicit constraints (SVOREX) or SVOR with implicit constraints (SVORIM) [47], have been proposed. Both methods are ordinal SVMs but with different constraints definition criteria in the optimization process. In the same way, an extended kernel discriminant learning [48] algorithm has been proposed to work as an ordinal classifier by using a ranking constraint. This method is known as kernel discriminant learning for ordinal regression (KDLOR) and has been proposed as an alternative to the ordinal SVM methods mentioned above.

The key difference between these approaches is that SVOR methods do not take advantage of the global information of the data (especially SVOREX which only considers adjacent classes when determining the thresholds) and also suffers from higher computational complexity.

Other approaches to ordinal classification, such as [49], rely on Gaussian processes (GPs) to develop an ordinal regressor: GP for ordinal regression (GPOR). The performance of this new method has been compared against SVORs techniques in several datasets, and the results showed a good generalization capacity and competitive performance. Furthermore, inspired by this approach and previous SVOR methods, Srijith et al. [50] have proposed a sparse GPOR using Leave-One-Out cross-validation to perform model selection (LOO-GPOR), a probabilistic least squares ordinal regressor (PLSOR) [51], and a semi-supervised ordinal regressor using GP [52]. Several recent works have been published in the literature, such as ORFEO [53], which focuses on specific problems where the ordinal output is derived from the original continuous output. ORFEO is an artificial neural network that simultaneously optimizes both outputs using a loss function that linearly combines ordinal classification and regression outputs. Marudi et al. [54] presented a novel decision tree-based method for ordinal classification that utilizes a generalization of the entropy measure for ordinal variables, along with an ordinal information gain ratio to assess the importance of each variable in the decision-making process.

DL for ordinal classification problems is an area that has been less explored up to now, with some works, such as [55] or [56], which studied the application of a new output layer based on CLMs together with a quadratic weighted kappa (QWK) loss function. This approach has been proposed for tackling ordinal image classification problems, for which excellent results are achieved in comparison to nominal approaches. A recent work [57] has taken this idea to approach the field of aesthetic quality control (AQC), developing a DL architecture with CLM as the output layer and cross-entropy as the cost function. The results have demonstrated the capacity of the network to take advantage of the ordinal nature of AQC problems, improving performance over previous nominal approximations. Furthermore, Rosati et al. [58] proposed two novel ordinal-hierarchical DL methodologies: 1) hierarchical conditional likelihood models, based on the CLM framework and 2) hierarchical-ordinal binary decomposition. Both approaches effectively model the ordinal structure across different hierarchical levels of the labels. Finally, [59] introduced a soft labeling approach based on generalized triangular distributions, which enables the model to minimize errors associated in distant classes on the ordinal scale.

At this point, once both main fields tackled in this work have been introduced, TSOC can be defined. TSOC consists of the application of ordinal classifiers to ordinal time-series problems, in such a way that the ordinal classifiers are able to consider and exploit the ordinal information present in the label space. TSOC is a recently established field only covered by a small set of conference papers [60], [61], [62]. The first two works have proposed an ordinal shapelet transformer in which the ordinal information of the data is exploited in two ways: 1) by introducing a novel shapelet quality measure in which ordinal information is taken into account and 2) by using an ordinal classifier instead of a nominal one, specifically the proportional odds model (POM) [63] and SVORIM [47] have been tested as final classifiers. Results have demonstrated that ordinal approaches significantly outperform nominal TSC approaches. Later, in [62], the effect of using different L_p norms for the computation of the shapelet quality has been studied. As can be seen, TSOC has room for significant improvement.

III. Preliminary Definitions

In this section, we provide formal definitions of TSOC concepts. Moreover, we also define a family of probabilistic ordinal classifiers (known as CLMs), which will be used for the TSOC methods proposed in this article.

A. Time Series and Related Supervised Learning Tasks

In supervised learning time-series problems, we are provided a training dataset, including a set of labeled time series, $D = \{(\mathbf{t}_1, y_1), (\mathbf{t}_2, y_2), \ldots, (\mathbf{t}_N, y_N)\}$, where N is the number of time series, $\mathbf{t}_i$ is the ith time series, y_i is the label assigned to it, and $i \in \{1, \ldots, N\}$. In general, the objective is to learn a mapping function (model) able to accurately predict the labels for the time series of the test set. Every time series is a set of C ordered sets (also known as channels) of real values. Although the most common type of time series are univariate ($C = 1$), there is an increasing interest in multivariate time series ($C > 1$). A multivariate time series with C channels is defined as $\mathbf{t}_i = (\mathbf{t}_i^1, \ldots, \mathbf{t}_i^C)$, where the cth channel is denoted as $\mathbf{t}_i^c = (t_{i,1}^c, t_{i,2}^c, \ldots, t_{i,T}^c)$, $c \in \{1, \ldots, C\}$, and T is the length of the time series. In this article, we focus on time series with a constant spacing of observation times, and we consider datasets where all the time series are equal length.

Depending on the nature of the labels, the supervised time-series problems can be categorized in TSER [64], [65], nominal TSC [13], or TSOC. TSER covers problems where the label to be predicted is a real number, i.e., $y_i \in \mathbb{R}$. For TSC, the label of each time series takes values in a discrete set of categories, $y_i \in \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_q, \ldots, \mathcal{C}_Q\}$, where $q \in \{1, \ldots, Q\}$, and Q is the number of classes of the problem. Finally, TSOC problems are TSC problems (with more than two classes) which include an order constraint in the labels.

B. Time-Series Ordinal Classification

The focus of this article is TSOC, where the categories show a natural order, in such a way that a relation $\mathcal{C}_1 \prec \mathcal{C}_2 \prec \ldots \prec \mathcal{C}_Q$ is present in label space. This relation is also present in TSER problems, where the $\prec$ operator is directly $<$ because we are working with real numbers instead of categories. However, the difference between TSOC and TSER is that the distance between categories is not known.

To provide a deeper understanding of this sort of problems, let us consider the example in Fig. 1. In this case, the classification problem involves determining the age ranges using the outlines of the middle finger. The ranges exhibit a natural relationship and can be categorized into three levels: 0–6 years old, 7–12 years old, and 13–19 years old. TSOC problems involve two important characteristics: 1) misclassification costs are different depending on the error (an error that confuses the 0–6 years old class with the 13–19 years old class should be penalized more heavily than one that confuses the 0–6 years old class with the 7–12 years old class) and 2) including the order information during learning should boost convergence.

1) Cumulative Link Models (CLMs): CLM belong to a broad family of models known as threshold-based models, which are built on two main ideas: 1) the existence of a real variable underlying the (discrete) output response, such that each class $\mathcal{C}_q$ belongs to a certain interval in $\mathbb{R}$. This hidden variable will be denoted as y^* and is often referred to in the literature as *latent variable* and 2) the use of a function f that transforms a pattern of the input space $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^D$ (where D is the input dimensionality) into the 1-D real space corresponding to the latent variable, $f : \mathbf{x} \in \mathbf{X} \subset \mathbb{R}^C \to \mathbb{R}$. The aim is to segment the real line into Q consecutive ordered intervals. Each interval along the real line corresponds to a specific class $\mathcal{C}_q$. These intervals are defined by a threshold vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{Q-1})$, where $\boldsymbol{\theta} \in \mathbb{R}^{Q-1}$. To guarantee that $P(y \preceq \mathcal{C}_q|\mathbf{x})$ increases with q, the thresholds vector must be nondecreasing, and, therefore, must satisfy the constraint $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_{Q-1}$ [63]. Thus, θ_1 is learned during the training process while the thresholds for $q = 2$ to $q = Q-1$ can be obtained as $\theta_q = \theta_1 + \sum_{i=2}^{q} \gamma_i^2$. $\boldsymbol{\gamma} = (\gamma_2, \gamma_3, \ldots, \gamma_{Q-1})$ is also learned during the training process. In this way, the aforementioned constraint is always satisfied. Considering this setting, an input $\mathbf{x}_i$ is associated with an output class $\mathcal{C}_q$ if $f(\mathbf{x}_i) \in [\theta_{q-1}, \theta_q]$. To completely cover the domain of the real line, θ_0 and θ_Q are set to $-\infty$ and $+\infty$, respectively, and are not considered part of the vector $\boldsymbol{\theta}$.

In view of the foregoing, the special case of CLM considers the latent variable model: $y^* = f(\mathbf{x}) + \epsilon$, where ϵ is the random error component with zero mean, $E(\epsilon) = 0$, where E represent the expected value, for which a certain distribution F_ϵ is assumed. Moreover, it assumes a linear latent variable $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. On that account, it is satisfied that

$$\begin{aligned} P(y \preceq \mathcal{C}_q|\mathbf{x}) &= P(y^* \leq \theta_q) \\ &= P(\mathbf{w}^T\mathbf{x} + \epsilon \leq \theta_q) \\ &= P(\epsilon \leq \theta_q - \mathbf{w}^T\mathbf{x}) \end{aligned} \tag{1}$$

taking g as the probability distribution function (p.d.f.) of the assumed F_ϵ, we get to

$$g(\theta_q - \mathbf{w}^T\mathbf{x}) = P(y \preceq \mathcal{C}_q|\mathbf{x}). \tag{2}$$

Therefore, the learning objective of the CLM is to find the set of optimal thresholds $\boldsymbol{\theta}^*$, as well as the optimal parameters $\mathbf{w}^*$ of the function f, in such a way that, given a pattern of the test set $(\mathbf{x}_i, y_i)$, the cumulative probabilities for each class $\mathcal{C}_q$, i.e., $P(y_i \preceq \mathcal{C}_q|\mathbf{x}_i)$, are as close as possible to the observed ones.

To this end, we need to define a loss function ℓ that represents the disagreement between the target label assigned to an input $\mathbf{x}_i$ and the output given by a prediction function $\text{pred}(\mathbf{p}_i)$, where $\mathbf{p}_i = (p_{i_1}, p_{i_2}, \ldots, p_{i_Q})$ is the vector of predicted probabilities, where $p_{i_q} = P(y_i = \mathcal{C}_q|\mathbf{x}_i) = P(y_i \preceq \mathcal{C}_q|\mathbf{x}_i) - P(y_i \preceq \mathcal{C}_{q-1}|\mathbf{x}_i)$. Hence, our objective is to find the function pred that minimizes the expected risk

$$\mathcal{L}(\text{pred}) = E(\ell(y_i, \text{pred}(\mathbf{p}_i)))$$

where $\mathcal{L}$ is the risk function. There are different options for the expected risk within the context of ordinal classification (e.g., the absolute difference in number of categories between the predicted and true labels). However, we cannot deal directly with this expected risk, mainly for two reasons: 1) the probability distribution that generates the patterns $(\mathbf{x_i}, y_i)$ of a dataset D is unknown and 2) the function ℓ is naturally discontinuous since its two arguments belong to the discrete space defined by the output y. This, as stated in [66], can lead to an NP-hard problem. Therefore, the common practice is to approximate ℓ by the so-called *surrogate risk*, which, considering the CLM setting, can be defined as

$$\mathcal{A}(f) = E(\psi(\boldsymbol{\theta}, f(\mathbf{x})))$$

where $\psi : \mathbb{R}^{Q-1} \times \mathbb{R} \to \mathbb{R}$ is generally referred to as the *surrogate loss* function. The need to introduce this surrogate terms is the main motivation for the existence of the previously introduced latent variable. As the values of y^* are not a priori known, we work with the vector $\boldsymbol{\theta}$, so that, for a pair $(\mathbf{x}_i, y_i)$, where y_i takes a value $\mathcal{C}_q$, we want the output of $f(\mathbf{x}_i)$ to be as close as possible to the θ_q threshold. To ensure the ordering of $\boldsymbol{\theta}$ introduced above, we essentially have two options. The first is to generate its elements through an incremental function, such as the one employed in the CLM activation layer (see Section III-B3). The second option is to design the cost function to naturally converge to a solution that maintains this ordering, as demonstrated by the logistic all-threshold (LogisticAT) method [66] used in our work.

2) Logistic All-Threshold: The LogisticAT method is a special case of CLM where the considered F_ϵ distribution is the logistic function. This leads to an interesting property that characterizes the so-called POMs. This property states that the ratio of the odds for two patterns inputs $\mathbf{x}_1$, $\mathbf{x}_2$ [67] is

$$\frac{\text{odds}(y \preceq \mathcal{C}_q|\mathbf{x}_1)}{\text{odds}(y \preceq \mathcal{C}_q|\mathbf{x}_2)} = \exp(-\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)).$$

The main distinguishing feature of LogisticAT with respect to the POM is its (*surrogate*) loss function, which is presented as

$$\begin{aligned} \psi_{\text{AT}} = \sum_{i=1}^{N} \left(\sum_{q=1}^{y_i - 1} h(\theta_q - \mathbf{w}^T\mathbf{x}_i) + \sum_{q=y_i}^{Q-1} h(\mathbf{w}^T\mathbf{x}_i - \theta_q) \right) \\ + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \end{aligned} \tag{3}$$

where $(\mathbf{x}_i, y_i)$ is a pattern of the training set defined in the form $h : \mathbb{R} \to \mathbb{R}$, $h(z) = \log(1 + \exp(z))$, $\mathbf{w}$ is the array of parameters associated with the function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ presented above, and λ is the regularization term, which is adjusted by cross-validation in our experiments (see Section V). Observe that in ψ_{AT}, by taking into account all the thresholds in $\boldsymbol{\theta}$ for each input $(\mathbf{x}_i, y_i)$, we ensure that at the point of convergence the ordering of $\boldsymbol{\theta}$ is satisfied.

This model is used for the ordinal convolutional-based techniques (O-ROCKET, O-Mini-Rocket, and O-MultiROCKET) discussed in Section IV-A.

3) CLM Activation Layer: The aforementioned mathematical structure of the CLM can be utilized as link function in DL architectures [55]. On this wise, the 1-D output of a deep network is mapped to a set of predicted probabilities through a CLM, for which we considered (as in the case of the LogisticAT method) a logistic distribution for the error component ϵ. In order to ensure the ordering of the vector $\boldsymbol{\theta}$, the next definition is considered: $\theta_q = \theta_1 + \sum_{i=1}^{q-1} \alpha_i^2$, with $q \in \{2, \dots, Q-1\}$, then θ_1 and $\alpha_2, \dots, \alpha_{Q-1}$ can be learned without constraints.

In addition, a cost function that takes greater account of ordinality is considered [68]. This function, denoted as ψ_{QWK}, is based on the QWK metric, and it is defined in terms of the predicted probabilities

$$\psi_{\text{QWK}} = \frac{\sum_{i=1}^{N} \sum_{q=1}^{Q} \omega_{y_i,q} P(y = \mathcal{C}_q | \mathbf{x}_i)}{\sum_{q=1}^{Q} \frac{N_q}{N} \sum_{j=1}^{Q} (\omega_{j,q} \sum_{i=1}^{N} P(y = \mathcal{C}_j | \mathbf{x}_i))}$$

where $\psi_{\text{QWK}} \in [0, 2]$, $(\mathbf{x}_i, y_i)$ is the ith sample of the data, N_q is the number of patterns labeled with the qth class, and $\omega_{j,q}$ are the elements of the penalization matrix, where $\omega_{j,q} = [(j-q)^2/(Q-1)^2]$.

This setting is implemented in the ordinal DL methodologies (O-InceptionTime, O-ResNet, O-LITETime, and O-CNN) discussed in Section IV-B.

IV. Proposed TSOC Methods

In this section, we introduce the seven TSOC methodologies. Within the category of convolutional-based techniques (Section IV-A), we present various adaptations to the ROCKET-family methods: O-ROCKET, O-MiniROCKET, and O-MultiROCKET. In addition, in line with DL techniques (Section IV-B), we introduce the O-InceptionTime, O-ResNet, O-LITETime, and O-CNN methodologies.

A. Convolutional-Based Techniques

This family of TSC algorithms was first proposed in [32] with the development of the ROCKET method for TSC. The main idea is to extract features from the input time series by applying a set of convolutional kernels. This convolution process involves a sliding dot product that depends on the properties of the kernel being applied, which are: the values of the kernels or weights ($\mathbf{w}$), the length (n), which sets the kernel extension (or number of weights); dilation (d), which is a popular technique to increase the length of the kernel without increasing the length of the resulting convolution. This is done by ignoring one out of every two values of the time series, effectively adding empty cells in the kernels [69]; padding (p), which adds to the beginning and end of the series a vector filled with zeros to control where the middle weight of the first and last kernel to be applied falls; and bias (b), a real value that is added to the kernel convolution result.

In addition to the original version of ROCKET, the authors subsequently presented two improved versions: 1) MiniROCKET [33] and 2) MultiROCKET [34]. They all share the same architecture, comprising four sequentially applied phases: 1) kernel convolution transform; 2) pooling operations; 3) standard scaler; and 4) the final LogisticAT classifier.

TABLE I
Differences Between O-ROCKET Approaches

	O-ROCKET	**O-MiniROCKET**	**O-MultiROCKET**
Kernel length	$\{7, 9, 11\}$	$\{9\}$	$\{9\}$
Weights	$\mathcal{N}(0, 1)$	{-1, 2}	{-1, 2}
Dilation	$\mathcal{U}(-1, 1)$	$\{2^0, \dots, 2^a\}$	$\{2^0, \dots, 2^a\}$
Use of Padding	random	alternative	alternative
Pooling operations	[PPV, GMP]	[PPV]	[PPV, GMP, MPV, MIPV, LSPV]
Num. Features	20,000	10,000	50,000

We propose the use of the LogisticAT method introduced in Section III-B1 as the final classifier, with a built-in cross validation of the λ regularization parameter. With respect to the standardization process and the final classifier, both remain unchanged. Conversely, the first two phases (kernel convolution and pooling operations), are different for each of the three alternatives. A comparison between the configuration of the three methods is presented in Table I.

1) O-ROCKET: ROCKET is a significant contribution to the state of the art in TSC, as it is capable of achieving excellent performance in a fraction of computational time. ROCKET applies to the input time series a large set of kernels generated with random properties: length n is randomly sampled from $\{7, 9, 11\}$ with equal probability; weights are sampled from a normal distribution $\forall w_k \in \mathbf{W}$, $w_k \sim \mathcal{N}(0, 1)$; bias b is sampled from a uniform distribution, $b \sim \mathcal{U}(-1, 1)$; dilation $d = \lfloor 2^x \rfloor, x \sim \mathcal{U}(0, A), A = \log_2([T-1]/[n-1])$ where T denotes the time-series length, and n the kernel length; and padding, which is denoted by p and is applied or not with the same probability in each kernel. In the case it is applied, its value is computed with $p = ((n-1) * d)/2$.

Once the kernel convolution is finished, two real-valued features are extracted from each kernel. The first is derived from global max pooling (GMP), which involves selecting the highest value. The second feature is the proportion of positives values (PPV), being PPV $= (1/n) \sum_{i=0}^{n-1} [z_i > 0]$, where z_i is the output of a single kernel convolution. Hence, PPV is a real value ranging from zero to one that represent the percentage of kernel convolutions that are greater than zero. These extracted features are then fed as input to the LogisticAT classifier.

2) O-MiniROCKET: MiniROCKET [33] is a later version of ROCKET which is up to 75 times faster than the original version, achieving similar performances. The main novelties with respect to ROCKET are the following, which make MiniROCKET become mostly (sometimes fully) deterministic.

1) The kernel length n is set to a fixed value of 9.
2) Kernel weights $\mathbf{w}$ are restricted to two possible values, -1 and 2.
3) A fixed dilation d is computed according to the input length. The d value can be in $\{2^0, \dots, 2^a\}$, where $a = \log_2(T-1)/(n-1)$.
4) Padding p is applied or not alternatively for each kernel/dilation combination. This way, half of the

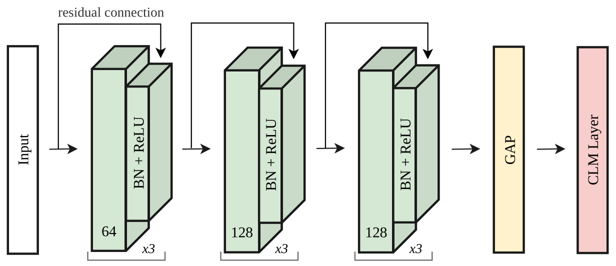


Fig. 2. O-ResNet architecture. The notation *x3* means that we have three stacked convolutional blocks inside each residual block. The final activation layer is the CLM (see Section III-B1).

kernels are computed with padding and the other half not. Whenever applied, its value is computed in the same way as in the O-ROCKET method.

5) Only the PPV aggregated feature is computed for each kernel.

3) O-MultiROCKET: This is the last version of the ROCKET family. In this novel variant, known as MultiROCKET [34], two main adjustments are made to MiniROCKET: 1) the convolution is also done on the first-order difference of the time series and 2) three additional aggregated features are computed: a) the mean positives values (MPVs); b) mean of indices of positives values (MIPV); and c) the longest stretch of positive values (LSPVs). These modifications substantially increase the computational expense of the algorithm but in turn achieves a significant increase in terms of accuracy.

B. Deep Learning Techniques

We propose four types of DL architectures: 1) the ordinal ResNet (O-ResNet) methodology based on the residual network architecture [37]; 2) the ordinal InceptionTime (O-InceptionTime) methodology based on the InceptionTime architecture [38]; 3) the ordinal light inception with boosting techniques (O-LITETime) based on the LITETime model [40]; and 4) the ordinal CNN (O-CNN) methodology, based on the CNN architecture [70].

1) O-ResNet: The original residual architecture was proposed in [71]. The main idea behind this sort of architectures was to add shortcut connections between nonadjacent layers, which facilitates the flow of the gradient through the network. This work was extended to the TSC paradigm in [37], in which the input time series is passed sequentially through a set of three residually connected convolutional blocks, which, for ease of understanding, are referred to as super-blocks (SBlocks). The reason for this notation is given by the fact that each super-block SBlock$_k$ is composed of three simple convolutional blocks with k filters each. In Fig. 2, we refer to this fact with the *x3* notation. In each SBlock$_k$, the convolution result is added to the original time series, and finally, a batch normalization layer (BN) and a rectified linear unit (ReLU) activation layer are applied. In our proposal, after the concatenated three SBlocks convolution, the final output of the network is obtained by applying a global average pooling (GAP) layer followed by a CLM activation layer (see Fig. 2).

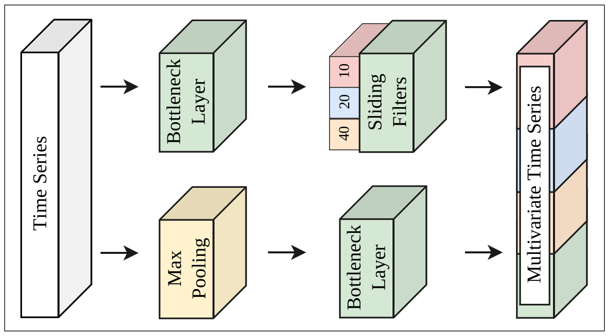


Fig. 3. IM architecture. Two pipelines are applied in parallel to the input time series. On the first one, the bottleneck layer is applied, followed by three sliding filter operations with sizes 40, 20, and 10, respectively. On the second one, a Max Pooling is performed, followed by another bottleneck layer.

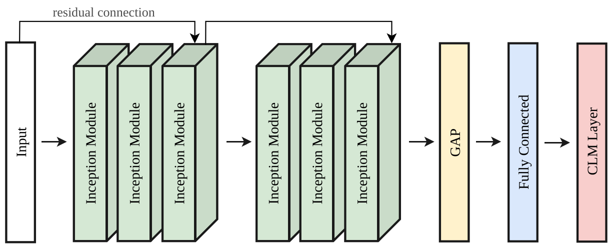


Fig. 4. O-IN architecture, two blocks of three IMs with residual connections are stacked, the resulting multivariate time series is passed to a GAP layer, followed by a fully connected and a CLM activation layer (see Section III-B1).

2) O-InceptionTime: The O-InceptionTime architecture takes some of the main ideas of the ResNet architecture and the original InceptionTime methodology [38], which includes concatenated convolution blocks together with residual connections between them. This method itself is an ensemble of five O-Inception Networks (O-INs). An O-IN comprises two residual blocks, each consisting of three IMs that remain unchanged from the initial InceptionTime proposal (see Fig. 3). O-IN applies a dimensionality reduction layer known as bottleneck, whose main goals are reducing the model complexity and avoiding overfitting. The convolution operation of a bottleneck layer consists in sliding m filters of length 1 with a stride equal to 1. Fig. 4 presents the O-IN architecture.

Specifically, an O-IN is composed of two residual blocks, each with three IMs. A GAP layer is applied to the output of the second block. Finally, the resulting features are fed to a fully connected layer, and finally, to the CLM activation layer.

As mentioned above, O-InceptionTime consists of five O-INs initialized with random weights, where the final prediction is performed by a majority voting system. This ensemble is built with the purpose of reducing the variability inherent to the ResNet architectures [72].

3) O-LITETime: This method, built upon the InceptionTime methodology, was developed to significantly reduce the computational complexity of its predecessor while maintaining competitive performance with other state-of-the-art methodologies. LITETime [40] achieves this balance by possessing only 2.34% of the number of parameters of InceptionTime. This reduction is primarily achieved through the use of bottleneck layers in the convolution process. To counteract this drastic simplification of the architecture, LITETime employs several boosting techniques: 1) *multiplexing*, which involves convolving the time series with kernels of different lengths simultaneously, allowing different convolution layers to be learned in parallel; 2) *dilation*, introduced in Section IV-A, which helps to more easily capture long-term patterns in the time series; and 3) *custom filters*, i.e., handcrafted kernels that facilitate the learning of specific patterns (typically difficult to learn) in the time series. Finally, after the convolution process, the final classification step is performed using a CLM layer, specifically adapted to ordinal classification.

4) O-CNN: This approach is an adaptation of the standard CNN methodology for 1-D data in the form of time series [70]. The CNN method was a significant contribution as it introduced the use of convolutional networks to the field of TSC. This methodology alternates between applying convolution and pooling layers to the input time series. In the convolution layers, kernels of lengths 5, 7, and 9 are used. For the pooling layers, dimensionality reduction is achieved by segmenting the convolution output and calculating the average of each segment. Finally, the result of the convolution is used to feed an MLP layer, coupled with a CLM layer, that performs the final classification.

V. Experimental Settings and Results

First, we present the methods used for the comparison. Then we define the first version of the UCO TSOC repository, which contains a total of 29 datasets. After that, the experimental setup is described. Finally, the results obtained by the seven proposed TSOC methods are compared with their nominal counterparts and with the rest of the nominal TSC approaches. Note that we provide open source `scikit-learn`-compatible implementations of all the methods benchmarked, a guide on how to reproduce experiments, and all the results obtained in an associated website.[2] All ordinal approaches will be available in the `aeon` toolkit[3] [73], which has also been used for benchmarking and analyzing the results obtained.

A. Methodologies Compared

To better understand how the proposed TSOC techniques perform, four different approaches have been run as a sanity check. The first two are a logistic regression [74] (LogReg) and an extreme gradient boosting (XGBoost) [75] classifier. For these, time series are flattened into a vector concatenating all the channels. Thus, a multivariate time series with C channels and length T is transformed into a single vector of length $C \times T$. The third baseline method is the time-series forest (TSF) [76]. Finally, the state-of-the-art approach in TSC, the HIVE-COTEv2 classifier (abbreviated as HC2) [77], has also been included to provide a deeper comparison. HC2 is an ensemble approach combining four approaches from different domains: 1) convolutional-based; 2) interval-based; 3) dictionary-based; and 4) shapelet-based. As explained above, these techniques are considered to enrich the experimentation, since they have been proven to be competitive in a wide range of problems.

B. TSOC Datasets

Given that we are covering the TSOC paradigm, all of the datasets considered must be ordinal in nature. Therefore, we made a selection from two well-known repositories, the UCR TSC repository,[4] from which we identified nine TSOC problems, and the Monash TSER repository,[5] from which 13 additional datasets have been chosen. In the latter case, as these are regression datasets, the original continuous output variable has been discretized into five uniformly distributed bins. Furthermore, five other datasets have been built from historical price data from five important companies in the stock market, and other two were formed from data collected from the National Data Buoy Center (NDBC) [78]. Hence, a total of 29 TSOC datasets have been considered in this work, with varying characteristics and backgrounds. This set of TSOC problems conforms the first version of the UCO TSOC repository[2]. Note that all the TSOC problems under consideration belong to the category of grouped continuous variables [29]. This is attributed to the fact that all the TSOC problems, sourced from diverse data repositories, entail the discretization of an underlying continuous variable. More information of the datasets, as well as specific information of the datasets, such as the number of classes, time-series length, the number of channels, and so on, is displayed in Appendix A in the supplementary material.

C. Experimental Settings

The seven novel TSOC methodologies detailed in Section III as well as the baseline approaches have been applied to the whole UCO TSOC repository. The datasets in their original form are initially divided into train and test partitions. Due to the stochastic nature of the methodologies and to mitigate the risk of overfitting to the default training data partition, each experiment is repeated 30 times under varied conditions. Specifically, every experiment is conducted using a different seed for the initialization of the methodologies and employing a distinct partition. For consistency, the first experiment is always executed with the default train/test partitions. Subsequent experiments, however, involve different partitions; the default train/test partitions are combined, and then 29 additional partitions are generated using a holdout procedure. Each of these partitions is created with a different seed (ranging from 1 to 29) while maintaining the same train/test proportion as the original dataset. We have to cross-validate only one hyperparameter in our experiments, λ, which is the regularization term of the LogisticAT classifier used in the ordinal convolutional-based methodologies. The set of λ values to be tested are obtained according to $10^{-3+(6i/9)}$, where $i \in \{0, 1, \ldots, 9\}$. A fivefold

[2]https://uco.es/ayrna/grupos/tsoc-dl-conv
[3]https://github.com/aeon-toolkit/aeon/
[4]https://timeseriesclassification.com
[5]http://tseregression.org/

stratified cross-validation approach is used. The best λ value is selected by mean absolute error (MAE), as it best represents the performance of ordinal approaches.

As this work deals with ordinal classification, specific ordinal metrics should be considered. Concretely, MAE, 1-Off accuracy (1-OFF) [79], and QWK [80] are considered. All these metrics aim to quantify how close the predictions are to the actual variable on the ordinal scale. In addition, the correct classification rate (CCR), widely used in nominal problems, is also computed.

D. Results

The first assessment is to determine whether the ordinal algorithms outperform their nominal counterparts. For this, the pairwise Wilcoxon signed-rank tests have been used with a significance level of 0.05. Note that ranks have been averaged across all datasets before conducting the statistical tests. The results are presented in Table II. Overall, TSOC methods are significantly better than nominal techniques. The exception is O-ResNet, where there is no discernible difference in any of the metrics.

The second experiment will determine which is the state-of-the-art approach in TSOC. Specifically, the seven ordinal versions of the proposed approaches are benchmarked along with the four comparative approaches introduced in Section V-A: XGBoost, LogReg, TSF, and HC2. For this, an adaptation of the critical difference diagram [81] has been used. The methodologies have been grouped into cliques, suggesting no significant difference in rank. These cliques were formed using the Holm correction for multiple testing as detailed in [82], with a significance level of 0.05. The results are graphically presented in Fig. 5, revealing that O-MiniROCKET consistently outperforms other methods across all metrics. Convolution-based methodologies, in general, demonstrate superior performance across all metrics. For MAE, O-MiniROCKET emerges as the top-performing approach, achieving significantly better results compared to all the other methods. Focusing now on the performance of the deep learners and keeping the convolution-based techniques aside, the O-InceptionTime is the best in terms of MAE and 1-OFF, being outperformed by O-CNN and by O-LITETime in terms of CCR and QWK, respectively. This is because deep learners require more training patterns to improve their performance. Notably, TSF and HC2 achieve their best performance for CCR, which aligns with their design objective. Moreover, except for MAE, where the O-MiniROCKET's superiority is statistically significant, and QWK, where both the O-MiniROCKET and O-MultiROCKET are the leading approaches (top clique), there are no significant differences among the top-performing methods for the remaining metrics. Therefore, it can be said that there is still room for better algorithms to be adapted and developed for TSOC problems. Furthermore, another way of improvement could be extending the problems archive, to include a wider range of datasets.

With the aim of evaluating the computational load of the methods, Fig. 6 compares the execution times in relation to the MAE. It is evident that ordinal ROCKET-based

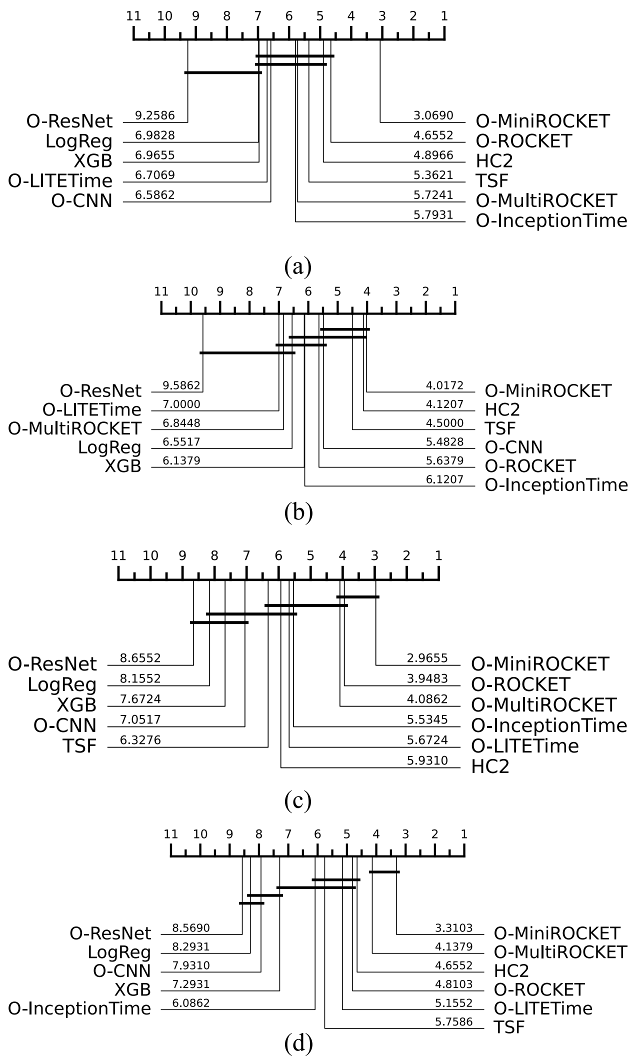


Fig. 5. Comparison between the TSOC methodologies and the baseline approaches described in Section IV. Methodologies are ordered based on the average rank over 30 resamples of train and test splits. (a) Results in terms of MAE. (b) Results in terms of CCR. (c) Results in terms of 1-OFF. (d) Results in terms of QWK.

methodologies exhibit slower performance compared to other TSOC approaches, such as O-ResNet or O-InceptionTime (run in GPU), but are faster than the HC2, the state-of-the-art nominal TSC approach. However, the O-ROCKET family methods outperform other methodologies, achieving over a 7% improvement in mean MAE compared to HC2 and over 10% against TSF, in the case of the O-MiniROCKET classifier. The increased computational load in the O-ROCKET-based methodologies is due to two main factors: 1) the extraction of kernels and computation of features and 2) the application of the final ordinal classifier. The former can be examined by comparing the different versions of ROCKET-based methods. For example, O-MiniROCKET is the fastest as it employs only one pooling operation, while O-ROCKET and O-MultiROCKET are slower as they apply two and five pooling operations, respectively, as specified in Table I. The latter aspect represents a potential area for optimization without compromising performance. Specifically, two potential enhancements for the ordinal versions of the O-ROCKET

TABLE II
p-VALUES OF THE WILCOXON PAIRED TESTS COMPARING TSC VERSUS TSOC VERSIONS OF THE METHODOLOGIES PROPOSED

	ROCKET vs O-ROCKET	MiniROCKET vs O-MiniROCKET	MultiROCKET vs O-MultiROCKET	ResNet vs O-ResNet	InceptionTime vs O-InceptionTime	LITETime vs O-LITETime	CNN vs O-CNN
MAE	**< 0.001**	**< 0.001**	0.156	0.442	**< 0.001**	0.305	**< 0.001**
CCR	0.053	0.086	0.304	0.831	**< 0.001**	1.000	0.139
QWK	**< 0.001**	**< 0.001**	**< 0.001**	0.381	0.345	**< 0.001**	0.203
1-OFF	**< 0.001**	**< 0.001**	**< 0.001**	0.155	**< 0.001**	**0.048**	**< 0.001**

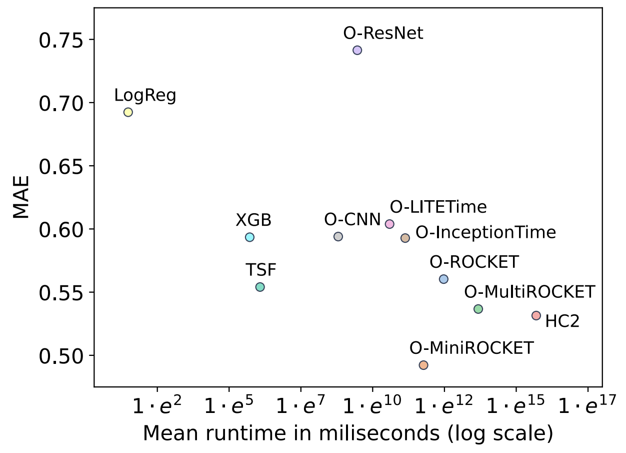


Fig. 6. Run time in milliseconds (log scale average over all problems) plotted against mean MAE in the test sets.

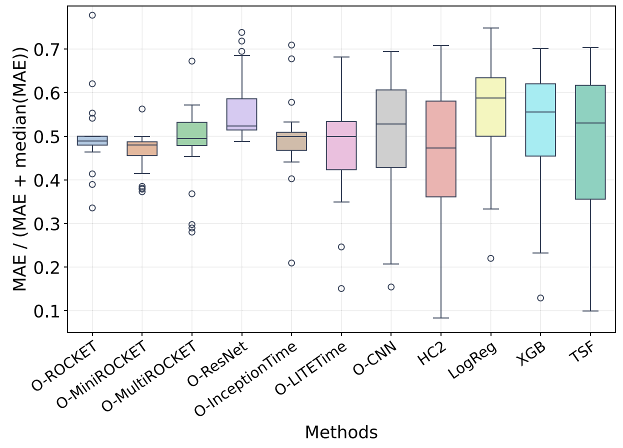


Fig. 7. Boxplot of relative MAEs.

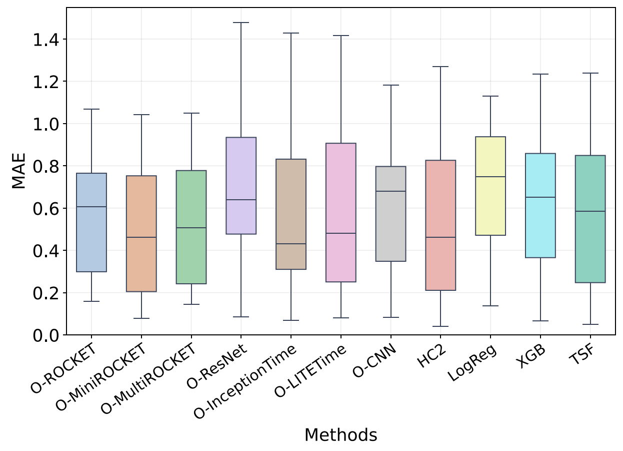


Fig. 8. Boxplot of the results obtained in MAE.

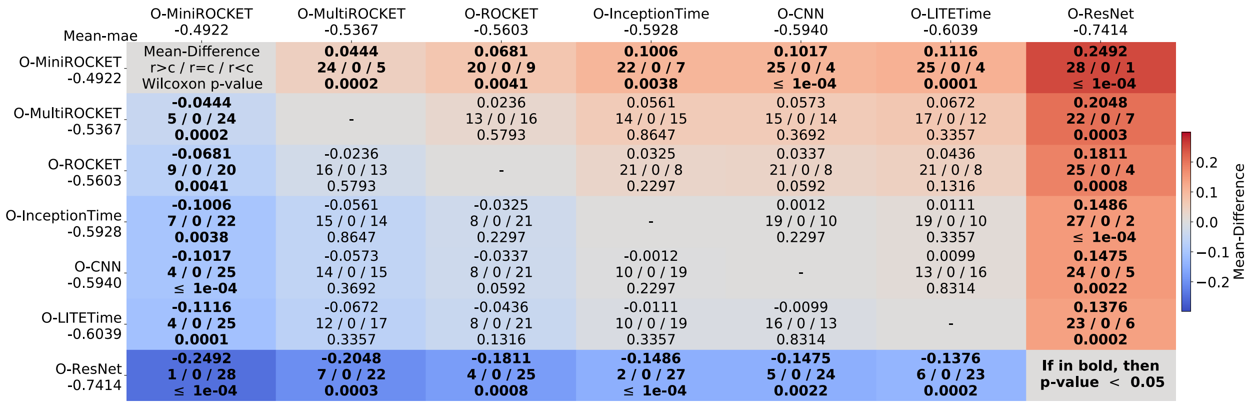
family methods include: 1) replacing the LogisticAT ordinal classifier with an ordinal version of the ridge classifier, which enables the computation of the projection matrix only once or 2) leveraging numba,[6] a just-in-time compiler for python. Both enhancements could lead to a significant improvement in computational time. A special mention should be given to TSF, which demonstrates a favorable MAE-computational time tradeoff, being orders of magnitude faster than ordinal O-ROCKET-based methodologies while achieving an acceptable mean MAE.

Additionally, the performance of each algorithm is compared in terms of relative MAE. The relative MAE is calculated by scaling the MAE of each approach with the median MAE for each dataset. Fig. 7 presents the relative MAE for each method using boxplots. In this figure, values greater than 0.5 indicate that the approach performs worse than the average method, while values smaller than 0.5 indicate that the approach performs better than the average method. It can be observed that the O-ROCKET family exhibits a larger spread in the values, with most values being smaller than 0.5. On the other hand, O-ResNet, O-CNN, the two standard approaches (LogReg and XGBoost), and the TSF approaches have values around 0.6, indicating that they perform worse than the average algorithm. To complement the analysis presented in Figs. 7 and 8 displays standard boxplots illustrating the complete distribution of the results. As observed, the majority of the results obtained by the ordinal convolutional approaches are consistently below a MAE of 1.0. The performance of the DL models is quite similar. Among the baseline nominal methodologies, HC2 stands out, demonstrating competitive performance relative to O-MiniROCKET. However, O-MiniROCKET consistently achieves the smallest Q1 and Q3 values and the smallest median value (on par with HC2), indicating its robustness and superior performance across the current TSOC repository.

Finally, in Fig. 9, a global comparison in terms of MAE between the presented ordinal techniques is provided in the form of a full pairwise multicomparison matrix (MCM) [83]. As can be observed, the MiniROCKET method is positioned as the best performing technique, obtaining a significant *p*-value (< 0.05), against the rest of techniques.

Finally, to provide a deeper understanding of the characteristics of the datasets for which the different techniques are

[6]https://numba.pydata.org/



Mean-mae	O-MiniROCKET -0.4922	O-MultiROCKET -0.5367	O-ROCKET -0.5603	O-InceptionTime -0.5928	O-CNN -0.5940	O-LITETime -0.6039	O-ResNet -0.7414
O-MiniROCKET -0.4922	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	**0.0444** **24 / 0 / 5** **0.0002**	**0.0681** **20 / 0 / 9** **0.0041**	**0.1006** **22 / 0 / 7** **0.0038**	**0.1017** **25 / 0 / 4** **≤ 1e-04**	**0.1116** **25 / 0 / 4** **0.0001**	**0.2492** **28 / 0 / 1** **≤ 1e-04**
O-MultiROCKET -0.5367	**-0.0444** **5 / 0 / 24** **0.0002**	-	0.0236 13 / 0 / 16 0.5793	0.0561 14 / 0 / 15 0.8647	0.0573 15 / 0 / 14 0.3692	0.0672 17 / 0 / 12 0.3357	**0.2048** **22 / 0 / 7** **0.0003**
O-ROCKET -0.5603	**-0.0681** **9 / 0 / 20** **0.0041**	-0.0236 16 / 0 / 13 0.5793	-	0.0325 21 / 0 / 8 0.2297	0.0337 21 / 0 / 8 0.0592	0.0436 21 / 0 / 8 0.1316	**0.1811** **25 / 0 / 4** **0.0008**
O-InceptionTime -0.5928	**-0.1006** **7 / 0 / 22** **0.0038**	-0.0561 15 / 0 / 14 0.8647	-0.0325 8 / 0 / 21 0.2297	-	0.0012 19 / 0 / 10 0.2297	0.0111 19 / 0 / 10 0.3357	**0.1486** **27 / 0 / 2** **≤ 1e-04**
O-CNN -0.5940	**-0.1017** **4 / 0 / 25** **≤ 1e-04**	-0.0573 14 / 0 / 15 0.3692	-0.0337 8 / 0 / 21 0.0592	-0.0012 10 / 0 / 19 0.2297	-	0.0099 13 / 0 / 16 0.8314	**0.1475** **24 / 0 / 5** **0.0022**
O-LITETime -0.6039	**-0.1116** **4 / 0 / 25** **0.0001**	-0.0672 12 / 0 / 17 0.3357	-0.0436 8 / 0 / 21 0.1316	-0.0111 10 / 0 / 19 0.3357	-0.0099 16 / 0 / 13 0.8314	-	**0.1376** **23 / 0 / 6** **0.0002**
O-ResNet -0.7414	**-0.2492** **1 / 0 / 28** **≤ 1e-04**	**-0.2048** **7 / 0 / 22** **0.0003**	**-0.1811** **4 / 0 / 25** **0.0008**	**-0.1486** **2 / 0 / 27** **≤ 1e-04**	**-0.1475** **5 / 0 / 24** **0.0022**	**-0.1376** **6 / 0 / 23** **0.0002**	**If in bold, then p-value < 0.05**

Fig. 9. MCM between ordinal methodologies in MAE. In each cell, three values are provided: 1) average of differences in MAE; 2) win/ties/losses counts; and 3) p-value of a Wilcoxon signed-rank test.

most suitable, the results have been analyzed based on the number of classes, the number of training patterns, time-series length, and the number of channels. This comparative analysis is presented in Appendix B in the supplementary material due to space constraints.

VI. Conclusion

This article presents the first application of convolutional- and DL-based techniques to TSOC, to the best knowledge of the authors. This area remains largely unexplored in comparison to nominal TSC. One of our contributions is the release of the UCO TSOC repository, including a total of 29 datasets from various domains. We also contribute to the literature with ordinal versions of two main categories in TSC: 1) convolutional- (ROCKET, MiniROCKET, and MultiROCKET) and 2) DL-based approaches (InceptionTime, CNN, ResNet, and LITETime). The ordinal versions of these techniques have resulted in significant performance improvements on the selected ordinal datasets against the nominal ones. Specifically, O-MiniROCKET outperforms all other approaches across all performance metrics, with significant differences in MAE, and does so with an acceptable computational time. Remarkably, even in terms of CCR, a nominal performance metric, O-MiniROCKET outperforms HC2, the state-of-the-art approach in nominal TSC. Overall, O-InceptionTime stands out as the most promising DL model, significantly improving upon the results of other deep learners. All results, source code, and guidance on reproducing experiments are available.

We recognize the potential for further advancements in TSOC algorithms. Incorporating alternative TSC techniques offers additional opportunities to explore this area and can serve as a baseline for future developments in this field. Another way of improvement is extending the experiments to develop ordinal counterparts of heterogeneous ensembles, such as HC2 [77]. It may be possible to adapt and apply this approach to TSOC with careful consideration. Another potential research direction in TSOC is multiobjective optimization, where two or more performance metrics may need to be simultaneously optimized, focusing on different aspects of ordinal classification [84].

Another area for improvement is in the UCO TSOC repository. Currently, the datasets are uniformly spaced, do not present missing values, and all time series are of equal length. However, we are conscious that many real-world TSOC datasets may not conform to these ideal conditions. Real-world TSOC problems often exhibit diverse characteristics, which may pose challenges, such as irregular spacing, presence of missing values, or variable lengths of time series. We believe that as the TSOC field matures, there will be opportunities to develop novel methodologies capable of accepting this diversity in input data characteristics. We also would like to extend our repository with new problems. We greatly appreciate any contributions to this archive.

Acknowledgment

Funding for open access charge: Universidad de Córdoba/CBUA.

References

[1] R. Khasha, M. M. Sepehri, and N. Taherkhani, “Detecting asthma control level using feature-based time series classification,” *Appl. Soft Comput.*, vol. 111, Nov. 2021, Art. no. 107694.

[2] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, “Financial time series forecasting with deep learning: A systematic literature review, 2005-2019,” *Appl. Soft Comput.*, vol. 90, 2020, Art. no. 106181.

[3] D. P. Roy and L. Yan, “Robust landsat-based crop time series modelling,” *Remote Sens. Environ.*, vol. 238, Mar. 2020, Art. no. 110810.

[4] G. Moody, “Spontaneous termination of atrial fibrillation: A challenge from PhysioNet and computers in cardiology,” in *Proc. Comput. Cardiol.*, vol. 31, 2004, pp. 101–104.

[5] J. Large, E. K. Kemsley, N. Wellner, I. Goodall, and A. Bagnall, “Detecting forged alcohol non-invasively through vibrational spectroscopy and machine learning,” in *Proc. 22nd Pacific-Asia Adv. Knowl. Discov. Data Min.*, 2018, pp. 298–309.

[6] Q. Liu et al., “Nonlinear spiking neural systems with autapses for predicting chaotic time series,” *IEEE Trans. Cybern.*, vol. 54, no. 3, pp. 1841–1853, Mar. 2024.

[7] M. Jin et al., “A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10466–10485, Dec. 2024.

[8] D. Guijo-Rubio, A. M. Durán-Rosal, P. A. Gutiérrez, A. Troncoso, and C. Hervás-Martínez, “Time-series clustering based on the characterization of segment typologies,” *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5409–5422, Nov. 2021.

[9] C. Holder, M. Middlehurst, and A. Bagnall, “A review and evaluation of elastic distance functions for time series clustering,” *Knowl. Inf. Syst.*, vol. 66, no. 2, pp. 765–809, 2024.

[10] N. M. Foumani, L. Miller, C. W. Tan, G. I. Webb, G. Forestier, and M. Salehi, “Deep learning for time series classification and extrinsic regression: A current survey,” *ACM Comput. Surv.*, vol. 56, no. 9, pp. 1–45, 2024.

[11] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Comput. Surv.*, vol. 57, no. 1, pp. 1–42, 2024.

[12] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Min. Knowl. Discov.*, vol. 31, no. 3, pp. 606–660, 2017.

[13] M. Middlehurst, P. Schäfer, and A. Bagnall, "Bake off redux: A review and experimental evaluation of recent time series classification algorithms," *Data Min. Knowl. Discov.*, vol. 38, pp. 1958–2031, Jul. 2024.

[14] A. Jastrzebska, G. Nápoles, W. Homenda, and K. Vanhoof, "Fuzzy cognitive map-driven comprehensive time-series classification," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 1348–1359, Feb. 2023.

[15] J. Mei, M. Liu, Y.-F. Wang, and H. Gao, "Learning a Mahalanobis distance-based dynamic time warping measure for multivariate time series classification," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1363–1374, Jun. 2016.

[16] X. Wan, L. Cen, X. Chen, Y. Xie, and W. Gui, "Memory shapelet learning for early classification of streaming time series," *IEEE Trans. Cybern.*, vol. 54, no. 5, pp. 2757–2770, May 2024.

[17] N. M. Foumani, C. W. Tan, G. I. Webb, and M. Salehi, "Improving position encoding of transformers for multivariate time series classification," *Data Min. Knowl. Discov.*, vol. 38, no. 1, pp. 22–48, 2024.

[18] H. Peng et al., "Reservoir computing models based on spiking neural P systems for time series classification," *Neural Netw.*, vol. 169, pp. 274–281, Jan. 2024.

[19] Á. Carmona-Poyato, N.-L. Fernández-García, F.-J. Madrid-Cuevas, R. Muñoz-Salinas, and F.-J. Romero-Ramírez, "Optimal online time-series segmentation," *Knowl. Inf. Syst.*, vol. 66, no. 4, pp. 2417–2438, 2024.

[20] P. Schäfer and U. Leser, "Motiflets: Simple and accurate detection of motifs in time series," *Proc. VLDB Endow.*, vol. 16, no. 4, pp. 725–737, 2022.

[21] P. Schäfer and U. Leser, "Discovering leitmotifs in multidimensional time series," 2024, *arXiv:2410.12293*.

[22] A. Jastrzebska, "Time series classification through visual pattern recognition," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 2, pp. 134–142, 2022.

[23] E. S. García-Treviño and J. A. Barria, "Structural generative descriptions for time series classification," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1978–1991, Oct. 2014.

[24] H. A. Dau et al., "The UCR time series archive," *IEEE-CAA J. Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, Nov. 2019.

[25] E. F. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 250–257.

[26] C. Li, Q. Liu, W. Dong, X. Zhu, J. Liu, and H. Lu, "Human age estimation based on locality and ordinal information," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2522–2534, Nov. 2015.

[27] D. Guijo-Rubio, P. A. Gutiérrez, C. Casanova-Mateo, J. Sanz-Justo, S. Salcedo-Sanz, and C. Hervás-Martínez, "Prediction of low-visibility events due to fog using ordinal classification," *Atmos. Res.*, vol. 214, pp. 64–73, Dec. 2018.

[28] K. Xu et al., "Application of ordinal logistic regression analysis to identify the determinants of illness severity of COVID-19 in China," *Epidemiol. Infect.*, vol. 148, p. e146, Jul. 2020.

[29] J. A. Anderson, "Regression and ordered categorical variables," *J. Roy. Stat. Soc. Ser. B, Methodol.*, vol. 46, no. 1, pp. 1–30, 1984.

[30] A. Bagnall and L. Davis, "Predictive modeling of bone age through classification and regression of bone shapes," 2014, *arXiv:1406.4781*.

[31] M. Shabani, D. T. Tran, J. Kanniainen, and A. Iosifidis, "Augmented bilinear network for incremental multi-stock time-series classification," *Pattern Recognit.*, vol. 141, Sep. 2023, Art. no. 109604.

[32] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Min. Knowl. Discov.*, vol. 34, pp. 1454–1495, Sep. 2020.

[33] A. Dempster, D. F. Schmidt, and G. I. Webb, "A very fast (almost) deterministic transform for time series classification," 2020, *arXiv:2012.08791*.

[34] C. W. Tan, A. Dempster, C. Bergmeir, and G. I. Webb, "MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification," *Data Min. Knowl. Discov.*, vol. 36, pp. 1623–1646, Sep. 2022.

[35] A. Dempster, D. F. Schmidt, and G. I. Webb, "Hydra: Competing convolutional kernels for fast and accurate time series classification," *Data Min. Knowl. Discov.*, vol. 37, no. 5, pp. 1779–1805, 2023.

[36] G. Uribarri, F. Barone, A. Ansuini, and E. Fransén, "Detach-ROCKET: Sequential feature selection for time series classification with random convolutional kernels," *Data Min. Knowl. Discov.*, vol. 38, pp. 3922–3947, Nov. 2024.

[37] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 1578–1585.

[38] H. I. Fawaz et al., "InceptionTime: Finding AlexNet for time series classification," *Data Min. Knowl. Discov.*, vol. 34, pp. 1936–1962, Nov. 2020.

[39] A. Ismail-Fawaz, M. Devanne, J. Weber, and G. Forestier, "Deep learning for time series classification using new hand-crafted convolution filters," in *Proc. IEEE Int. Conf. Big Data*, 2022, pp. 972–981.

[40] A. Ismail-Fawaz, M. Devanne, S. Berretti, J. Weber, and G. Forestier, "LITE: Light inception with boosting techniques for time series classification," in *Proc. IEEE 10th Int. Conf. Data Sci. Adv. Anal.*, 2023, pp. 1–10.

[41] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognit. Lett.*, vol. 42, pp. 11–24, Jun. 2014.

[42] Q. Ma, Z. Chen, S. Tian, and W. W. Ng, "Difference-guided representation learning network for multivariate time-series classification," *IEEE Trans. Cybern.*, vol. 52, no. 6, pp. 4717–4727, Jun. 2022.

[43] Y. Ren, L. Li, X. Yang, and J. Zhou, "Autotransformer: Automatic transformer architecture design for time series classification," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Min.*, 2022, pp. 143–155.

[44] L. Bai et al., "HAQJSK: Hierarchical-aligned quantum Jensen–Shannon kernels for graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6370–6384, Nov. 2024.

[45] M. Li et al., "Guest editorial: Deep neural networks for graphs: Theory, models, algorithms, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4367–4372, Apr. 2024.

[46] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal regression," in *Proc. 9th Int. Conf. Artif. Neural Netw. (ICANN)*, 1999, pp. 97–102.

[47] W. Chu and S. Keerthi, "Support vector ordinal regression," *Neural Comput.*, vol. 19, pp. 792–815, Mar. 2007.

[48] D. Olsson, P. Georgiev, and P. M. Pardalos, "Kernel principal component analysis: Applications, implementation and comparison," in *Proc. 2nd Int. Conf. Netw. Anal.*, vol. 59, 2013, pp. 127–148.

[49] W. Chu and Z. Ghahramani, "Gaussian processes for ordinal regression," *J. Mach. Learn. Res.*, vol. 6, no. 35, pp. 1019–1041, 2005.

[50] P. K. Srijith, S. Shevade, and S. Sundararajan, "Validation based sparse Gaussian processes for ordinal regression," in *Proc. Int. Conf. Neural Inf. Process.*, 2012, pp. 409–416.

[51] P. Srijith, S. Shevade, and S. Sundararajan, "A probabilistic least squares approach to ordinal regression," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2012, pp. 683–694.

[52] P. K. Srijith, S. Shevade, and S. Sundararajan, "Semi-supervised Gaussian process ordinal regression," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2013, pp. 144–159.

[53] A. M. Gómez-Orellana, D. Guijo-Rubio, P. A. Gutiérrez, C. Hervás-Martínez, and V. M. Vargas, "ORFEO: Ordinal classifier and regressor fusion for estimating an ordinal categorical target," *Eng. Appl. Artif. Intell.*, vol. 133, Jul. 2024, Art. no. 108462.

[54] M. Marudi, I. Ben-Gal, and G. Singer, "A decision tree-based method for ordinal classification problems," *IISE Trans.*, vol. 56, no. 9, pp. 960–974, 2024.

[55] V. M. Vargas, P. A. Gutierrez, and C. Hervas-Martinez, "Cumulative link models for deep ordinal classification," *Neurocomputing*, vol. 401, pp. 48–58, Aug. 2020.

[56] V. M. Vargas, P. A. Gutiérrez, and C. Hervás-Martínez, "Deep ordinal classification based on the proportional odds model," in *Proc. Int. Work-Conf. Interplay Between Nat. Artif. Comput.*, 2019, pp. 441–451.

[57] R. Rosati, L. Romeo, V. M. Vargas, P. A. Gutiérrez, C. Hervás-Martínez, and E. Frontoni, "A novel deep ordinal classification approach for aesthetic quality control classification," *Neural Comput. Appl.*, vol. 34, pp. 11625–11639, Jul. 2022.

[58] R. Rosati, L. Romeo, V. M. Vargas, P. A. Gutierrez, E. Frontoni, and C. Hervas-Martinez, "Learning ordinal–hierarchical constraints for deep learning classifiers," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 12, 2024, doi: 10.1109/TNNLS.2024.3360641.

[59] V. M. Vargas, A. M. Durán-Rosal, D. Guijo-Rubio, P. A. Gutiérrez, and C. Hervás-Martínez, "Generalised triangular distributions for ordinal deep learning: Novel proposal and optimisation," *Inf. Sci.*, vol. 648, Nov. 2023, Art. no. 119606.

[60] D. Guijo-Rubio, P. A. Gutierrez, A. Bagnall, and C. Hervas-Martinez, "Time series ordinal classification via shapelets," in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–8.

[61] D. Guijo-Rubio, P. A. Gutiérrez, A. Bagnall, and C. Hervás-Martínez, "Ordinal versus nominal time series classification," in *Proc. Int. Workshop Adv. Anal. Learn. Tempor. Data*, 2020, pp. 19–29.

[62] D. Guijo-Rubio, V. M. Vargas, P. A. Gutiérrez, and C. Hervás-Martínez, "Studying the effect of different L_p norms in the context of time series ordinal classification," in *Proc. Conf. Spanish Assoc. Artif. Intell.*, 2021, pp. 44–53.

[63] P. McCullagh, "Regression models for ordinal data," *J. Roy. Stat. Soc., Ser. B, Methodol.*, vol. 42, no. 2, pp. 109–127, 1980.

[64] C. W. Tan, C. Bergmeir, F. Petitjean, and G. I. Webb, "Time series extrinsic regression: Predicting numeric values from time series data," *Data Min. Knowl. Discov.*, vol. 35, no. 3, pp. 1032–1060, 2021.

[65] D. Guijo-Rubio, M. Middlehurst, G. Arcencio, D. F. Silva, and A. Bagnall, "Unsupervised feature based algorithms for time series extrinsic regression," *Data Min. Knowl. Discov.*, vol. 38, pp. 2141–2185, Jul. 2024.

[66] F. Pedregosa, F. Bach, and A. Gramfort, "On the consistency of ordinal regression methods," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 1769–1803, 2017.

[67] P. Gutiérrez, M. Pérez-Ortíz, J. Sánchez-Monedero, F. Fernández-Navarro, and C. Hervás-Martínez, "Ordinal regression methods: Survey and experimental study," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 127–146, Jan. 2016.

[68] J. de La Torre, D. Puig, and A. Valls, "Weighted kappa loss function for multi-class classification of ordinal data in deep learning," *Pattern Recognit. Lett.*, vol. 105, pp. 144–154, Apr. 2018.

[69] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.

[70] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *J. Syst. Eng. Electron.*, vol. 28, no. 1, pp. 162–169, 2017.

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[72] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep neural network ensembles for time series classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2019, pp. 1–6.

[73] M. Middlehurst et al., "aeon: A python toolkit for learning from time series," *J. Mach. Learn. Res.*, vol. 25, no. 289, pp. 1–10, 2024.

[74] D. R. Cox, "The regression analysis of binary sequences," *J. Roy. Stat. Soc., Ser. B, Methodol.*, vol. 20, no. 2, pp. 215–232, 1958.

[75] T. Chen and T. He. "XGBoost: Extreme gradient boosting." XGBoost. 2015. [Online]. Available: https://cran.r-project.org/web/packages/xgboost/vignettes/xgboost.pdf

[76] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inf. Sci.*, vol. 239, pp. 142–153, Aug. 2013.

[77] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "HIVE-COTE 2.0: A new meta ensemble for time series classification," *Mach. Learn.*, vol. 110, pp. 3211–3243, Dec. 2021.

[78] A. Gómez, D. Guijo-Rubio, P. Gutiérrez, and C. Hervás-Martínez, "Simultaneous short-term significant wave height and energy flux prediction using zonal multi-task evolutionary artificial neural networks," *Renew. Energy*, vol. 184, pp. 975–989, Jan. 2022.

[79] J.-C. Chen, A. Kumar, R. Ranjan, V. M. Patel, A. Alavi, and R. Chellappa, "A cascaded convolutional neural network for age estimation of unconstrained faces," in *Proc. 8th Int. Conf. Biom. Theory, Appl. Syst.*, 2016, pp. 1–8.

[80] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[81] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

[82] S. García and F. Herrera, "Extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, no. 89, pp. 2677–2694, 2008.

[83] A. Ismail-Fawaz et al., "An approach to multiple comparison benchmark evaluations that is stable under manipulation of the comparate set," 2023, *arXiv:2305.11921*.

[84] Z. Gong, H. Chen, B. Yuan, and X. Yao, "Multiobjective learning in the model space for time series classification," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 918–932, Mar. 2019.

Rafael Ayllón-Gavilán received the B.Sc. degree in computer science in 2021 and the M.Sc. degree in artificial intelligence from University of Córdoba, Córdoba, Spain, in 2022. He is currently pursuing the Ph.D. degree in computer science with the University of Córdoba, Córdoba, Spain.

His current research concerns the time-series classification domain, addressing the ordinal case together with its possible real-world applications.

David Guijo-Rubio (Member, IEEE) received the Ph.D. degree in computer science from the University of Córdoba, Córdoba, Spain, in 2021.

He is currently working as an Assistant Professor with the University of Córdoba. His current interests include different tasks applied to time series (classification—nominal and ordinal; clustering; and regression).

Pedro Antonio Gutiérrez (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Granada, Granada, Spain, in 2009.

He is currently a Full Professor with the University of Córdoba, Córdoba, Spain. His research interests are supervised learning and ordinal classification.

Prof. Gutiérrez serves on the editorial board of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Anthony Bagnall received the Ph.D. degree in computer science from the University of East Anglia, Norwich, U.K., in 2001.

He is currently a Full Professor with the University of Southampton, Southampton, U.K. His primary research interest is in time-series machine learning, with focus on classification, but more recently looking at clustering and regression. He has a side interest in ensemble design.

César Hervás-Martínez (Senior Member, IEEE) received the Ph.D. degree in mathematics from the University of Seville, Seville, Spain, in 1986.

He is currently a Full Professor of Computer Science with the University of Córdoba, Córdoba, Spain, leading the AYRNA Research Group. His current research interests include neural networks, evolutionary computation, and the modeling of natural systems.