A Classification Strategy for Internet of Things Data Based on the Class Separability Analysis of Time Series Dynamics

JOÃO B. BORGES, Universidade Federal do Rio Grande do Norte, Brazil and Universidade Federal de Minas Gerais, Brazil
HEITOR S. RAMOS and ANTONIO A. F. LOUREIRO, Universidade Federal de Minas Gerais, Brazil

This article proposes TSCLAS, a time series classification strategy for the Internet of Things (IoT) data, based on the class separability analysis of their temporal dynamics. Given the large number and incompleteness of IoT data, the use of traditional classification algorithms is not possible. Thus, we claim that solutions for IoT scenarios should avoid using raw data directly, preferring their transformation to a new domain. In the ordinal patterns domain, it is possible to capture the temporal dynamics of raw data to distinguish them. However, to be applied to this challenging scenario, TSCLAS follows a strategy for selecting the best parameters for the ordinal patterns transformation based on maximizing the class separability of the time series dynamics. We show that our method is competitive compared to other classification algorithms from the literature. Furthermore, TSCLAS is scalable concerning the length of time series and robust to the presence of missing data gaps on them. By simulating missing data gaps as long as 50% of the data, our method could beat the accuracy of the compared classification algorithms. Besides, even when losing in accuracy, TSCLAS presents lower computation times for both training and testing phases.

CCS Concepts: • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies** → *Machine learning*; • **Mathematics of computing** → *Information theory;*

Additional Key Words and Phrases: Internet of Things, time series classification, time series dynamics, ordinal patterns transformation, Bandt-Pompe transformation, class separability index

ACM Reference format:
João B. Borges, Heitor S. Ramos, and Antonio A. F. Loureiro. 2022. A Classification Strategy for Internet of Things Data Based on the Class Separability Analysis of Time Series Dynamics. *ACM Trans. Internet Things* 3, 3, Article 23 (July 2022), 30 pages.
https://doi.org/10.1145/3533049

This work was supported in part by research agencies CAPES, CNPq, FAPEMIG, and in part by São Paulo Research Foundation (FAPESP), under Grants #15/24494-8, #18/23064-8, and #2020/05121-4.
Authors' addresses: J. B. Borges, Universidade Federal do Rio Grande do Norte, Department of Computing and Technology, Caicó, RN, 59300-000, Brazil and Universidade Federal de Minas Gerais, Department of Computer Science, Belo Horizonte, MG, 30123-970, Brazil; email: joao.borges@ufrn.br; H. S. Ramos and A. A. F. Loureiro, Universidade Federal de Minas Gerais, Department of Computer Science, Belo Horizonte, MG, 30123-970, Brazil; emails: {ramosh, loureiro}@dcc.ufmg.br.

2577-6207/2022/07-ART23 $15.00
https://doi.org/10.1145/3533049

1 INTRODUCTION

The classification of time series is a fundamental research topic that has gained a new breath in the last years, mainly due to the increasing needs from big data and Internet of Things (IoT) communities in better understanding the data they are dealing with [32, 49, 54]. A considerable number of strategies was already proposed for the task of time series classification in general [3]. However, for the challenging scenario of IoT, the use of traditional methods is not always possible [9].

For instance, to process IoT data from different sources, one must consider handling large time series data generated at different rates, of different types and magnitudes, possibly having issues concerning uncertainty, inconsistency, and incompleteness due to missing readings and sensor failures [9, 17, 26, 37, 49]. These problems directly affect data quality and, consequently, make it harder, or even impracticable, to apply traditional classification methods to them [8]. For instance, missing readings create gaps within the time series, which may require a prior imputation step on the raw data [57]. However, it can be impracticable as the gap increases, affecting the performance and scalability of classification methods. Furthermore, the randomness of missing readings create time series with different lengths, while most classifiers assume time series of equal length.

In this work, we propose TSCLAS, a time series classification strategy for the IoT data, based on the class separability analysis of their temporal dynamics. Since TSCLAS must be suited for IoT scenarios, in addition to the requirements for the classification task itself, it is designed to be robust to the following challenges: (i) the large length and (ii) incompleteness of IoT data. For both challenges, we follow the claim that solutions to handle data from IoT sensors should avoid using raw data, considering their transformation to another representation domain [8]. To consider the first challenge, we follow a fast and scalable method to transform the large time series in a small set of features, reducing the processing time of a further classification strategy. Consequently, for the second challenge, we show that, with the proper transformation on the raw data, we are able to mitigate the problems derived from the incompleteness of data, and propose a more robust time series classification approach.

Our proposal is based on the ordinal patterns transformation, a representation domain for the characterization of data according to its dynamics, indicating how a system evolves over time [7, 43]. By considering its dynamics, it is possible to evaluate its data generating process, giving us the ability to capture the inner characteristics of the underlying phenomena, which is a feasible strategy for dealing with the issues of IoT data. The ordinal patterns transformation, proposed by Bandt and Pompe [5], is a powerful tool to evaluate time series dynamics, that has been extensively discussed and applied to real-world studies [1, 2, 40, 44, 45]. Furthermore, this transformation focuses on the simplicity and the fast calculation, being invariant to differences in magnitudes, which are essential aspects of our scenario. In summary, the ordinal patterns transformation, which is detailed in Section 3, consists of transforming a given time series in a set of ordinal patterns, according to two parameters: an embedding dimension D and an embedding delay τ. Roughly speaking, the parameter D is used for constructing sliding windows of size D, which will define the ordinal patterns, and the parameter τ corresponds to the timescale interval used to sample the consecutive points of those sliding windows. However, despite the transformation's ability for the characterization and distinction of time series dynamics [42, 44], when applied to real-world challenging data, which is the case for IoT, some aspects must be considered in order to increase its classification potential. An essential aspect is, thus, the choice of transformation parameters (D,τ), which is an important step in the classification process.

The importance of choosing the best parameters for the ordinal patterns transformation is clear, but most studies in the literature use off-the-shelf values. For instance, the choice of the best embedding dimension D is still an open question, where the current solutions follow the

recommendations from literature [5], by choosing $D \in \{3, \ldots, 7\}$. The main aspect to consider here is that the chosen D depends on the time series length n, satisfying the condition that $n \gg D!$. On the other hand, for the embedding delay, a common choice is to assume $\tau = 1$. However, although these values are enough for a controlled scenario, it may impact the classification results for the challenging IoT data. In this work, we investigate the selection of these parameters, discussing the main conditions for the embedding dimension D and providing a strategy for finding the most appropriate embedding delay τ.

Thus, for given a train/test dataset, TSCLAS performs a strategy for selecting the best parameters for the ordinal patterns transformation, based on the maximization of the class separability of the time series dynamics within the training dataset. It is important to mention that this is not a hyperparameter tuning approach, where the best parameters are defined by their classification results assessed on a different validation set. Instead, we only consider the training dataset and its classification potential given the separation of their classes for different parameters. Our class separability analysis is performed on the causality complexity-entropy plane (CCEP) [42, 44]. This plane consists of the normalized permutation entropy and the statistical complexity, as the x-axis and y-axis, respectively, both computed from the ordinal patterns probability distribution. It was successfully applied as a method for distinguishing different time series dynamics, by placing each type of dynamics at different locations [1, 2, 42, 44]. We show that the best distinction of time series dynamics within the CCEP occurs whenever there is a clear separability between the time series classes, since the distribution of their points does not intersect with others. Thus, our strategy consists of choosing the best parameters that maximize the class separability of the distribution of points for the classes on the plane.

Thus, once the parameters are chosen, TSCLAS is applied for the classification of IoT data, based on a combination of features extracted from the ordinal patterns probability distribution and the ordinal patterns transition graph, both derived transformations from the set of computed ordinal patterns. As shown in the following, it is scalable to the length of time series and robust to missing data gaps, common issues from IoT context not properly covered by traditional classification strategies.

In summary, in this work, we present the following contributions:

(1) a class separability measure for time series dynamics, according to their displacement on the CCEP, indicating its potential for correctly distinguishing the time series,
(2) a method for choosing the best parameters for the ordinal patterns transformations according to the maximization of the separability measure of time series classes, and
(3) a time series classification strategy for IoT data, which is scalable to time series length and robust to missing gaps, through features extracted from the ordinal patterns transformations.

The rest of this article is organized as follows. Section 2 discusses studies related to our proposed methods, focusing on class separability strategies and the use of the ordinal patterns transformations in other problem domains. Section 3 briefly describes the concepts of ordinal pattern transformations and the set of features computed from them, which are needed in our classification strategy. Section 4 presents our proposed class separability measure for time series dynamics, and Section 5 introduces TSCLAS, our strategy for using these measures in the time series classification. Section 6 shows our experiments and results. Finally, Section 7 presents our final conclusions and future directions.

2 RELATED WORK

The top-rank state-of-art solutions for time series classification consist of general-purpose classifiers, independently of the type and application domain of the time series. One of the first proposals

with significant results classification that remains relevant nowadays is the time series forest (TSF) classifier [14], which is an ensemble algorithm that improves the well-established **random forest (RANDF)** method. The use of an ensemble of classifiers is a common basis for these strategies, which is the case of the **Bag-of-SFA-Symbols (BOSS)** [48], the ensemble of **elastic distances (EE)** [23], and **Random Interval Spectral Ensemble (RISE)** [24] classifiers. Following this approach, the solutions with the most significant results, but that also take the ensemble method to its extreme, is the **COllective of Transformation-based Ensembles (Flat-COTE)** [4], with 35 classifiers in its ensemble, followed by its **hierarchical version, the HIVE-COTE** [25]. However, most of these strategies require a considerable processing time for training its classifiers, even for small datasets, which are not adequate for IoT scenarios. Consequently, a simpler classification algorithm, such as the k-nearest neighbors (KNN), is a feasible solution for scenarios with large time series. The interested reader might refer to the "bake of" work of Bagnall et al. [3] for more details regarding these classifiers.

For the challenging IoT time series, unlike the general-purpose classifiers, the proposed solutions for classifying IoT data are very domain-specific, making them hard to reproduce in different contexts. For instance, Montori et al. [33] proposed a classification strategy considering a scenario where some textual information to describe IoT data is provided. However, not all sensors have a description available, making this approach unfeasible. Another aspect usually considered when classifying IoT data, as surveyed by Wu [57], consists of strategies that are robust to their issues, mainly the data incompleteness. In this direction, Postol et al. [36] proposed a strategy for the classification of noisy and incomplete IoT data based on **topological data analysis (TDA)**. However, although their solution achieves good results, their proposal is well suited for time series that spans over months, with the best results with one month for training and eight months for testing. Although not directly related to the time series classification task, some strategies try to be robust to data imperfections in the realm of industrial IoT time series. Chen et al. [13] proposed a strategy for predicting future values of time series even in the presence of missing data, based on a variational inference algorithm for a **kernel dynamic Bayesian network (KDBN)**. In the same direction, Bekiroglu et al. [6] proposed a strategy for predicting the output measure of a sensor by combining multiple inputs, which can handle the case when there are imperfections in these inputs.

In a previous work, Borges et al. [8], we proposed a strategy for the identification of IoT time series according to their dynamical behavior. In that work, we evaluated each behavior based on their placement in the **causality complexity-entropy plane, the CCEP**, which is constructed after the ordinal patterns transformation. However, some issues with this method impact the correct distinction of time series dynamics and their accuracy in classification results. For instance, the separation of classes within CCEP are not clear for $\tau = 1$ when the time series are noisy or does not satisfy the method's conditions. Thus, our novel contribution with the present work follows the studies of Zunino et al. [61] and Borges et al. [7], which show that it is possible to discover important characteristics of the time series when evaluating the extracted metrics as a function of τ. In another previous work, Borges et al. [7], we demonstrated that a strategy considering a multiscale approach, i.e., evaluating metrics from ordinal patterns transformations as a function of τ, is suitable for scenarios with short time series and does not depend on selected parameters. However, this approach requires successive transformations on the time series, each of them with a different value of $\tau \in \{1, \ldots, 50\}$, which is impracticable given the large numbers of IoT data. In this direction, our proposal consists of selecting the best parameters for transforming the ordinal patterns before the classification based on the class separability of the time series dynamics.

The separation of classes is a classical question that arises in many distinct areas. From molecular biology [29, 55] to astronomy [28, 60], whenever there is a graphical representation of points that belong to different classes, there is the need to understand to what extent we can separate a particular subset from the others [12, 56]. The class separability is a measure used to identify the regions of different classes, which can be used to distinguish them [12, 53]. It consists of geometric analysis of the space where points are distributed, which tries to find regions or boundaries that separate each class [53]. This generalizes the linear separability concept. The Thornton's Geometric Separability Index (GSI) [53] was one of the first defined separability measures. It is defined as the proportion of data points whose classification labels are the same as those of their nearest neighbors, accounting for the degree to which inputs associated with the same output tend to cluster together [16]. This method has shown that it is possible to empirically analyze the predictability at the core of the learning methods. This led to novel class separability measures that have been proposed for different problems [12, 28, 29, 55, 56, 60].

In this work, we apply this concept of class separability analysis to the graphical presentation of time series dynamics from the CCEP. However, contrary to most class separability indices in the literature, which are based on some distance measure to identify different classes, distances in the CCEP are not easy to consider. For instance, given the minimum and maximum limits for statistical complexities in the plane, as illustrated in Section 4, a straight line between two points is not always possible, since it may not respect those limits. Thus, we took another path to propose the separability index of TSCLAS, which is based on the estimation of classes regions and the intersection among them.

3 ORDINAL PATTERNS TRANSFORMATIONS AND METRICS

Bandt and Pompe [5] proposed a novel representation for time series based on the ordinal relation between successive data points [2, 61]. Their methodology transforms the time series into a set of patterns, which can capture the ordinal structure of time series, thus the ordinal name. These ordinal patterns are related to the temporal correlation of the time series underlying phenomena, being a valuable indicator of the time series dynamics [22, 42, 44].

This representation has been extensively discussed in recent studies, where it was shown to be well suited for real-world data [1, 2, 40, 44, 45]. Furthermore, it was proposed to focus on simplicity and fast calculation, and robust to observational and dynamical noises. These are all important properties for our application scenario, which will be exploited by TSCLAS. In the following, we present more details to perform the ordinal patterns transformations in time series data, their parameters, and features employed to characterize the dynamics of such time series. The code of all algorithms presented here is available at a public repository.[1]

3.1 Ordinal Patterns Transformation

Consider a time series $\mathbf{x} = x_1, \ldots, x_n$ of length n and an embedding dimension $D \in \mathbb{N}$. The first step of the Bandt-Pompe method requires the generation of sliding windows $\mathbf{w}_t \subseteq \mathbf{x}$ of size D, such that, for each instant $t = 1, \ldots, n - (D - 1)$,

$$\mathbf{w}_t = x_t, x_{t+1}, \ldots, x_{t+(D-2)}, x_{t+(D-1)}. \tag{1}$$

Thus, for each window $\mathbf{w}_t$, at a given instant t, the ordinal pattern consists of the permutation $\pi_t = (r_1, r_2, \ldots, r_D)$ of $(1, 2, \ldots, D)$ such that

$$x_{t+r_1-1} \leq x_{t+r_2-1} \leq \cdots \leq x_{t+r_{D-1}-1} \leq x_{t+r_D-1}. \tag{2}$$

[1]Implementations of the Bandt-Pompe ordinal patterns transformations: https://github.com/labepi/bandt_pompe.

In other words, π_t represents the necessary permutation for the elements of $\mathbf{w}_t$ to be sorted in ascending order. A proposal that advances this method consists of inserting an embedding delay $\tau \in \mathbb{N}$ [2, 42], such that the elements within each sliding window can be separated by intervals of size τ, corresponding to a sample of the time series by regular spaced intervals [61]. Thus, the sliding windows for each instant t are defined as

$$\mathbf{w}_t = x_t, x_{t+\tau}, \ldots, x_{t+(D-2)\tau}, x_{t+(D-1)\tau}. \tag{3}$$

The method, originally proposed by Bandt and Pompe [5], is equivalent to the case where $\tau = 1$, and the sliding windows are sampled by consecutive data samples. Consequently, they are comprised of τ-spaced data points for $\tau > 1$.

After the transformation, the time series is converted into the set of ordinal patterns $\Pi = \{\pi_i : i = 1, \ldots, n-(D-1)\tau\}$, and each π_i represents a permutation from the set of $D!$ possible permutations. The choice of D depends on the length n of the time series, and the condition $n \gg D!$ must be satisfied to obtain reliable statistics. For practical purposes, D is in the interval 3 and 7 [61].

Algorithm 1 illustrates the steps necessary to perform the transformation of a given time series $\mathbf{x}$ of size n to its ordinal patterns Π. The time complexity of this algorithm is $O(nD^2)$, assuming that the permutation is obtained by sorting each sliding window by the function `Order()` in Line 4, which can be a simple sorting algorithm such as the selection sort. However, for practical purposes, D is recommended to be a small value ($D \in \{3, \ldots, 7\}$) [5], and, thus, the sorting algorithm will take at most 7 elements, and the complexity of this strategy depends mainly on the size n of the time series, which is $O(n)$ in practice.

ALGORITHM 1: OrdinalPatterns

Input: The time series $\mathbf{x}$ of size n, the embedding dimension D, and the embedding delay τ.
Output: The sequence of ordinal patterns Π.

```
// The sequence of ordinal patterns
1 Π ← {π_i : i = 1, ..., n − (D − 1)τ} ;
2 for t ← 1 to n − (D − 1)τ do
      // Get the sliding window w at t
3     w_t ← x_t, x_{t+τ}, ..., x_{t+(D−2)τ}, x_{t+(D−1)τ};
      // The current permutation pattern index
4     π_i ← Order(w_t);
5 return Π;
```

3.2 Ordinal Patterns Probability Distribution

Given the set Π of ordinal patterns obtained from the transformation of a given time series of length n, the ordinal patterns probability distribution p_π consists in assigning a probability distribution to the permutations identified in the time series. Thus, for each possible permutation $\pi_t \in \Pi$, with $t = 1, \ldots, D!$, let $|s_{\pi_t}| \in \{0, \ldots, m\}$ be the number of observed patterns of type π_t. The ordinal patterns probability distribution $p_\pi = \{p(\pi_t) : \forall t = 1, \ldots, D!\}$ is defined as

$$p(\pi_t) = \frac{|s_{\pi_t}|}{n-(D-1)\tau}, \tag{4}$$

satisfying the conditions $p(\pi_t) \geq 0$ and $\sum_{\pi_t} p(\pi_t) = 1$.

Algorithm 2 illustrates the steps to compute the ordinal patterns probability distribution p_π from a given time series $\mathbf{x}$ of length n. Since the number of ordinal patterns from the time series obtained after the ordinal patterns transformation is $|\Pi| = n-(D-1)\tau$, and $n \gg D!$, this algorithm's time complexity is bounded by $O(n)$.

ALGORITHM 2: OrdinalPatternsPD

Input: The sequence of ordinal patterns Π, and the embedding dimension D.
Output: The ordinal patterns probability distribution p_π.

```
// The distribution of permutations
p_π ← {p_{π_i} = 0 : i = 1, ..., D!} ;
for i ← 1 to |Π| do
    // Counting the pattern frequency
    p_{π_i} ← p_{π_i} + 1/|Π|;
return p_π;
```

3.3 Ordinal Patterns Transition Graph

Recent approaches for time series representations lie at the crossing between ordinal patterns and graph representations from time series. They are based on the transformation of time series into networks according to their ordinal patterns [18, 31, 50, 59]. These approaches consider each $D!$ possible ordinal pattern as a vertex in the graph.

Given the sequence of ordinal patterns Π, the ordinal patterns transition graph G_π represents the relations between consecutive ordinal patterns, and is defined as a directed weighted graph $G_\pi = (V, E)$ with $V = \{\pi_i : i = 1, \ldots, D!\}$, where each vertex corresponds to one of the $D!$ possible permutations for an embedding dimension D, and a set of directed weighted edges $E = \{(\pi_i, \pi_j) : \pi_i, \pi_j \in V\}$. A directed edge connects two ordinal patterns in the graph if they appear sequentially in the time series, representing a transition between them, thus the name "ordinal patterns transition graph". There is an edge $(\pi_i, \pi_j) \in E$, between the vertices π_i and π_j, if there is a sequence of ordinal patterns $\Pi_t = \pi_i$ and $\Pi_{t+1} = \pi_j$, $1 \leq t \leq n - (D-1)\tau - 1$. The weight function $w : E \to \mathbb{R}$ of an edge represents the probability of a specific transition in Π. Thus, the weight of a given edge (π_i, π_j) is given by

$$w(\pi_i, \pi_j) = \frac{|\Pi_{\pi_i, \pi_j}|}{m - 1}, \tag{5}$$

where $|\Pi_{\pi_i, \pi_j}| \in \{0, \ldots, n - (D-1)\tau - 1\}$ is the number of transitions between permutations π_i and π_j, with $\sum_{\pi_i, \pi_j \in V} w(\pi_i, \pi_j) = 1$.

Once this graph representation is constructed, some studies use unweighted edges [18, 31] to represent only the existence of such transitions. However, in this work, we follow the approach to consider the relative frequency of transitions as the weights of edges [50, 59].

Algorithm 3 presents the steps to compute the ordinal patterns transition graph G_π from the set of ordinal patterns Π obtained from a given time series $\mathbf{x}$ of length n. The graph G_π is represented here as an adjacency matrix of order $D! \times D!$. The same analysis of time complexity made for the Algorithm 2 can be performed for this case. Thus, this algorithm's time complexity is bounded by $O(n)$, since it directly depends on the number of ordinal patterns $|\Pi| = n - (D-1)\tau$, and $n \gg D!$. For more details and examples regarding the ordinal patterns transition graph, please refer to [7].

3.4 Features from the Ordinal Patterns Transformations

To perform a proper characterization of time series dynamics from the previously discussed ordinal patterns transformations, we follow the literature, which considers the extraction of relevant features from those transformations [1, 2, 15, 40, 42, 44, 45]. To measure their capacity of enhancing important aspects from the underlying time series, such features must be chosen to improve the distinction between those time series, which is not clear when using only their raw data.

In this work, we evaluate a set of features extracted from both the ordinal patterns probability distribution p_π and the ordinal patterns transition graph G_π, described in the following sections.

ALGORITHM 3: OrdinalPatternsTG

Input: The sequence of ordinal patterns Π, and the embedding dimension D.
Output: The ordinal patterns transition graph G_π.

```
// The adjacency matrix representation
```
$G_\pi \leftarrow \{g_{\pi_i,\pi_j} = 0 : i = 1, \ldots, D! \text{ and } j = 1, \ldots, D!\}$;
for $i \leftarrow 2$ **to** $|\Pi|$ **do**
```
    // Counting the transition patterns frequencies
```
    $g_{\pi_{i-1},\pi_i} \leftarrow g_{\pi_{i-1},\pi_i} + 1/(|\Pi| - 1)$;
return G_π

3.4.1 Features from p_π. From p_π, we compute the following features:

(1) the normalized permutation entropy ($H_S[p_\pi]$),
(2) the statistical complexity ($C_{JS}[p_\pi]$), and
(3) the Fisher information measure ($F[p_\pi]$).

These are information theory quantifiers that have already been used as significant measures for the proper characterization of the underlying behavior of time series [1, 2, 15, 19, 40, 42, 51, 58].

Permutation entropy. Following the initial purpose of Bandt and Pompe [5], the information quantifiers are calculated from the distribution p_π for all $D!$ permutations π of order D.

Since p_π is defined on the permutations of neighboring values, the authors proposed a variation from the classical entropy of Shannon, which they called *permutation entropy*, and is defined as

$$H[p_\pi] = -\sum_{t=1}^{D!} p(\pi_t) \log p(\pi_t), \tag{6}$$

where $0 \leq H[p_\pi] \leq \log D!$. The permutation entropy is equivalent to the Shannon entropy and is a measure of uncertainty associated to the process described by p_π [2]. Lower values of $H[p_\pi]$ represent an increasing or decreasing sequence of values in the permutation distribution, indicating that the original time series is deterministic. On the other side, high values of $H[p_\pi]$ indicate a completely random system [5].

Normalized Permutation Entropy. The maximum value for $H[p_\pi]$ occurs when all $D!$ possible permutations have the same probability of occurring, which is the case for the uniform distribution p_u of permutations. Thus, $H_{\max} = H[p_u] = \log D!$, where $p_u = \{1/D!, \ldots, 1/D!\}$ [61].

Rosso et al. [42] defined the normalized Shannon entropy, from the permutation entropy case, as

$$H_S[p_\pi] = \frac{H[p_\pi]}{H_{\max}}, \tag{7}$$

where $0 \leq H_S[p_\pi] \leq 1$.

Statistical Complexity. Another statistical measure that can be computed from the ordinal patterns probability distribution p_π is the statistical complexity. While the entropy gives the notion of the uncertainty of a system, spanning the extremes of perfect predictable to completely randomness, the statistical complexity is used to represent the uncertainty between those extremes. Defined by Lamberti et al. [21], this measure presents a different perspective regarding the knowledge of some underlying process, capturing the relationship between the dynamical components of a system, such as determinism and randomness, giving the idea of the disequilibrium between them.

Based on the Jensen-Shannon divergence JS, detailed in Lamberti et al. [21], between the associated probability distribution p_π and the uniform distribution p_u, i.e., the trivial case for the minimum knowledge from the process, the statistical complexity is given by

$$C_{JS}[p_\pi] = Q_{JS}[p_\pi, p_u]H_S[p_\pi], \tag{8}$$

where $p_\pi = \{p(\pi)\}$ is the ordinal patterns probability distribution, p_u is the uniform distribution, and $H_S[p_\pi]$ is the normalized Shannon entropy, as previously described.

The disequilibrium $Q_{JS}[p_\pi, p_u]$ is given by

$$Q_{JS}[p_\pi, p_u] = Q_0 JS[p_\pi, p_u] \tag{9}$$

$$= Q_0 \left\{ S\left[\frac{p_\pi + p_u}{2}\right] - \frac{S[p_\pi] + S[p_u]}{2} \right\}, \tag{10}$$

where S is the Shannon entropy measure. Q_0 is a normalization constant, given by

$$Q_0 = -2 \left\{ \left(\frac{D! + 1}{D!} \right) \ln(D! + 1) - 2\ln(2D!) + \ln(D!) \right\}^{-1}, \tag{11}$$

which is equal to the inverse of the maximum value of $JS[p_\pi, p_u]$, so $0 \leq Q_{JS} \leq 1$ [2, 46].

Fisher information. Following the characterization of distinct dynamics from the ordinal patterns probability distribution p_π, the Fisher information is a measure able to capture the concentration of a given distribution. Contrary to the Shannon entropy, which provides a notion of the uncertainty of a system by measuring the global spreading of the distributions, the Fisher information is considered to have a locality property because it considers the differences among consecutive probabilities within the distribution [43, 47].

The Fisher information for the discrete case is given by

$$F[p_\pi] = F_0 \sum_{t=1}^{D!-1} \left(\sqrt{p_{t+1}} - \sqrt{p_t}\right)^2, \tag{12}$$

where F_0 is a normalization constant defined as

$$F_0 = \begin{cases} 1 & \text{if } p_{i^*} = 1 \text{ for } i^* = 1 \text{ or } i^* = N \text{ and } p_i = 0, \forall i \neq i^*, \\ 1/2 & \text{otherwise.} \end{cases} \tag{13}$$

By measuring these local changes, the Fisher information can provide a quantifier of the gradient of the distribution, which increases as the density is more concentrated [47]. For instance, for a distribution with density concentrated at a single value, $F[p_\pi] \approx 1$, while $H[p_\pi] \approx 0$.

3.4.2 Features from G_π. From G_π, we extracted the following set of features:

(1) number of edges (N_E)
(2) mean of edges weights (μ_{E_w});
(3) standard deviation of edges weights (σ_{E_w});
(4) normalized Shannon entropy of edges weights distribution ($H_S[E_w]$);
(5) statistical complexity of edges weights distribution ($C_{JS}[E_w]$);
(6) Fisher information of edges weights distribution ($F[E_w]$); and
(7) probability of self transitions (p_{st}).

Once we have the graph representation of the transition between patterns, there are several features to be computed. For instance, Ravetti et al. [39] constructed their metrics from the nodes degree distribution, but this requires a large number of vertices to be relevant, which are not our intention, since this leads to the need for a large D, increasing the computational execution time.

Instead, following the studies of Zhang et al. [59], Borges et al. [7], and Olivares et al. [34], we focus on the features that represent the transitions themselves, which are related to the graph edges.

Number of Edges. Since the edges of G_π represent the occurrence of transitions between consecutive patterns, the number of edges is an important indicator of the time series dynamics. For instance, as the randomness of a process increases, it also increases the chances for all possible transitions in the graph to occur. The number of edges is computed by

$$N_E = |E[G_\pi]|. \tag{14}$$

Mean and Standard Deviation of Edges Weights. The edges from G_π are defined to represent the probability of the existence of a specific transition in Π. Thus, as described in Equation (5), the weights induce a probability distribution $p_{E_w} = \{w(\pi_i, \pi_j) : \forall i, j = 1, \ldots, D!\}$. Consequently, the mean and standard deviation from p_{E_w} are given by

$$\mu_{Ew} = \frac{1}{N_E} \sum w(\pi_i, \pi_j), \tag{15}$$

and

$$\sigma_{Ew} = \sqrt{\frac{1}{N_E - 1} \sum w(\pi_i, \pi_j)}. \tag{16}$$

Information Theory Quantifiers from Edges Weights Distribution. Similarly to the features computed from p_π, we calculate the same information theory quantifiers from edges weights of G_π. These are the normalized Shannon entropy of edges weights ($H_S[E_w]$), the statistical complexity of edges weights ($C_{JS}[E_w]$), and the Fisher information of edges weights ($F[E_w]$).

Probability of Self-Transitions. The probability of self-transitions is defined by Borges et al. [7] as the probability of the occurrence of a sequence of equal patterns within the set of ordinal patterns, and can be expressed as

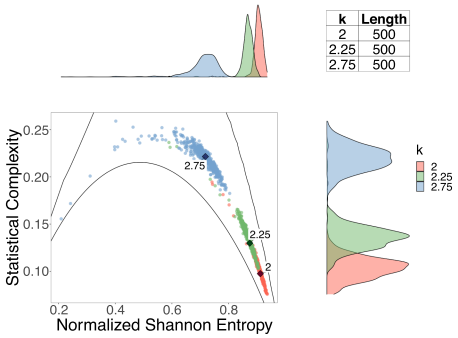
$$p_{st} = p(\pi_i, \pi_i) = \sum_{i \in \{1, \ldots, D!\}} w(v_{\pi_i}, v_{\pi_i}). \tag{17}$$

The self-transitions, which are represented as the loops in the graph, are directly related to the temporal correlation of time series, and are a valuable indication of their underlying dynamics. In fact, Borges et al. [7] showed that the self-transitions, when evaluated as a function of the embedding delay τ, represent a valuable indicator of the main characteristics of the time series, even for small values of D, and independently on a single value of τ.

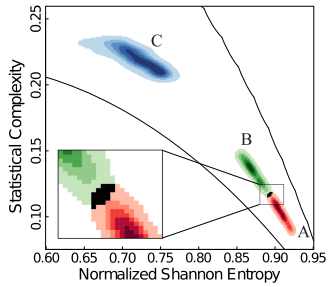
4 THE CLASS SEPARABILITY OF TIME SERIES DYNAMICS

A powerful tool for the characterization of time series dynamics is the causality complexity-entropy plane (CCEP), proposed by Rosso et al. [42, 44]. The CCEP is a plane formed by the normalized permutation entropy (H_S) and the statistical complexity (C_{JS}) measures, computed from the ordinal patterns probability distribution p_π, as the x-axis and the y-axis of the plane, respectively. It was successfully applied as a method to distinguish different time series dynamics. For instance, it can be used to identify noise from chaos, which is a significant achievement given the intriguing aspect that time series from chaotic systems share several properties with stochastic processes, making their distinction a hard task [20, 39, 40, 42–44, 61].

According to the (H_S, C_{JS}) pair, different time series dynamics are expected to be placed at specific regions of the plane. These regions are also bounded by minimum and maximum limits of the statistical complexity. More details regarding several types of dynamic behavior, such as noisy, chaotic, or deterministic behaviors and their expected placements within the CCEP can be found in Rosso et al. [42], Ravetti et al. [39], and Borges et al. [7]. However, although this method can



(a) CCEP with points placements according to its k and their marginal distributions along the H_S and C_{JS} axes.



(b) Transformed grid from the CCEP of Figure 1a, after filtering the most significant points and estimating the classes distributions with KDE ($n_{KDE} = 250$). Intersection area is highlighted.

Fig. 1. Illustration of CCEP, for $D = 4$ and $\tau = 1$, of time series generated as colored random noises, defined by their power spectra f^{-k} [22], representing three equal sized classes with $k \in \{2, 2.25, 2.75\}$. Continuous lines represent the minimum and maximum limits for statistical complexities.

distinguish most of these dynamics, some conditions must be satisfied by the time series data, so it can be placed at the expected position.

First, to assure the reliability of the computed measures, the time series length n must be long enough so the sampling in the $D!$ space of patterns is representative, i.e., $n \gg D!$ [44, 61]. If this condition is not satisfied, it may lead to a statistical analysis that is not consistent with the real-time series behavior and the occurrence of missing patterns [41]. Furthermore, even if the length is satisfied, in real-world scenarios, data may not be pure concerning a single dynamic behavior. A dominant behavior may be corrupted by spurious noise or even by rounding or truncating values, which may affect the precision when measuring their dynamics. These issues add a certain degree of uncertainty to the measures, impacting their placement in the CCEP.

Figure 1(a) shows a CCEP, for $D = 4$ and $\tau = 1$, illustrating the placements of $1,500$ colored random noise time series, each of them with $n = 1{,}000$ samples, and defined by their power spectra f^{-k} [22], representing three equal sized classes with $k \in \{2, 2.25, 2.75\}$. Classes are represented as red, green, and blue points, respectively, for each k (represented by the highlighted diamond points in the figure). Continuous lines represent the minimum and maximum limits for statistical complexities Their marginal density distributions also illustrates the spread of points along H_S and C_{JS} axes.

It can be noted that the potential for correctly distinguishing the time series dynamics with the CCEP is related to the placement distribution of points for each class in the plot. Assuming the distribution of points in Figure 1(a), the points of the class $k = 2.75$ are more straightforward to distinguish than the others since the distribution of their points does not intersect with others. Otherwise, the intersection between classes for $k = 2$ and $k = 2.25$ makes their distinction harder since there is no clear separability between those time series classes, which, however, may lead to misclassifications. Thus, to measure this potential of classification with the CCEP, let us define class separability as follows:

Definition 4.1 (Class Separability). Let $\mathcal{D}$ be a given dataset composed of a number of time series, with each of them labeled from a set of classes $C = \{c_1, \ldots, c_l\}$. The class separability for $\mathcal{D}$, projected over a CCEP, with pairs (H_S, C_{JS}) computed with D and τ, corresponds to the ratio between the intersected areas from the distribution of points and the non-intersected areas.

To analyze the class separability for this scenario, we have to define our version of a separability index S_I for the classes of a given dataset. Before describing S_I, let us establish some requirements:

(1) It should be robust to outliers to reduce the effects of noise and imprecision of the measurements on the analysis.
(2) It should be non-parametric, given the placement area of points in CCEP, it is not easy to fit *a priori* distribution for the data to calculate their intersection ranges.
(3) It should be independent of the number of instances per class, and, thus, we must consider the area covered by the points in a broad sense, and not computing the index per point.

These requirements are essential due to the possible expensive computational cost that quickly arises when dealing with the magnitudes of IoT data. So, as a solution for requirement (1), we propose using a method for removing the outliers by computing the points by their concentration range. Any method could be applied here, but we follow the approach described in Borges et al. [8], which achieved good results by using the Skinny-dip [30] clustering strategy on the points for each class. Skinny-dip is a noise-robust clustering algorithm based on the Hartigan's dip test of modality, which can reasonably detect the more distinguished concentration of points in a given region. In this work, we use a statistical significance level of $\alpha = 0.05\%$ for estimating the clusters.

For requirements (2) and (3), we propose estimating class distributions, after filtering the most significant points, by a kernel density estimation (KDE). For our case, it is sufficient to apply a two-dimensional KDE given our bivariate case with the H_S and C_{JS} axes of CCEP. As a result, KDE will transform the CCEP onto a $n_{KDE} \times n_{KDE}$ square grid. A large n_{KDE} impacts on the precision of the estimated regions and has implications on the processing time. Figure 1(b) shows the combined version of the resulting grids for each class from the CCEP of Figure 1(a), after applying the methods above with $n_{KDE} = 250$.

Let us model these grids as $n_{KDE} \times n_{KDE}$ matrices $\mathbf{A} = \{a_{i,j}\}$, $\mathbf{B} = \{b_{i,j}\}$, and $\mathbf{C} = \{c_{i,j}\}$, corresponding to the areas covered by the classes distributions formed by points from colored noises with $k = 2$, $k = 2.25$, and $k = 2.75$, respectively. Without loss of generality, let us consider the case of the class distribution from $k = 2$ modeled by $\mathbf{A}$. Each one of their elements is defined as $a_{i,j} = d_{i,j}$, with $i, j \in \{1, \ldots, n_{KDE}\}$, where $d_{i,j}$ represents the values according to the estimated density of points at that location from its underlying CCEP.

Thus, for each ijth element, we want to measure the proportion of the density this element of $\mathbf{A}$ has concerning other distributions intersecting at these positions. Then, each $a_{i,j}$ element is normalized as

$$a'_{i,j} = \frac{a_{i,j}}{\sum_{c \in C} c_{i,j}}, \tag{18}$$

where $c \in C$ is used as a correspondence for all the classes of labels in the given dataset.

Consequently, the possible values of $a'_{i,j}$ are

$$a'_{i,j} = \begin{cases} 0 & \text{if } a_{i,j} = 0, \\ 1 & \text{if } c_{i,j} = 0, \forall c \in C \text{ and } c \neq a, \\ ]0, 1[& \text{otherwise.} \end{cases} \tag{19}$$

Note that the proportion $0 < a'_{i,j} < 1$ only occurs in case of intersection at that point.

After the normalization of all values, having a normalized matrix $\mathbf{A}'$, we can compute the individual separability s_c for each class $c \in C$. Let $A' = \{a'_{i,j} | a'_{i,j} > 0\}$ be the set of elements of $\mathbf{A}'$ with non-zero values. If $m = |A'|$, then, the individual separability for a given class A is given by

$$s_A = \frac{\sum_i \sum_j a'_{i,j}}{m}. \tag{20}$$

The following lemmas are directly derived from the equations above.

Lemma 4.2. *For a given class distribution without intersection with any other class, its separability achieves the maximum value equals* 1.

Proof. If a normalized grid $\mathbf{A}'$, transformed from a given class distribution has no intersections, then, $a'_{i,j} = 1, \forall i, j \in \{1, \ldots, n_{KDE}\}$. Thus, its individual separability is given by

$$s_A = \frac{\sum_i \sum_j 1}{m} = \frac{m}{m} = 1. \tag{21}$$

□

Lemma 4.3. *The minimum individual separability for a given class distribution occurs when all l classes distributions are completely overlapped, and it is equal to* $1/l$.

Proof. Let l be the number of class distributions in a given dataset. If all classes are perfectly overlapped, then their estimated grids will be equally distributed over the CCEP. So, all elements of a given normalized grid $\mathbf{A}'$ have intersections with all other classes, with the same density values and, consequently, $a'_{i,j} = 1/l, \forall i, j \in \{1, \ldots, n_{KDE}\}$. Thus, its individual separability is given by

$$s_A = \frac{\sum_i \sum_j 1/l}{m} = \frac{m/l}{m} = 1/l. \tag{22}$$

□

Finally, given the individual separability for all classes in C, the separability index S_I is given by

$$S_I = \frac{1}{l} \sum_{c \in C} S_c. \tag{23}$$

The bounds for the class separability index S_I is presented by Theorem 4.4.

Theorem 4.4 (Bounds of Separability Index). *The separability index S_I for a given dataset $\mathcal{D}$, composed of a number of time series labeled from a set of classes $C = \{c_1, \ldots, c_l\}$, with respect to its projected CCEP, is bounded by* $1/l \leq S_I \leq 1$.

Proof. Following Lemmas 4.2 and 4.3, the maximum value of the separability index S_I of a given dataset occurs when all classes $c \in C$ achieve their maximum individual separabilities s_c. Conversely, the minimum S_I occurs when s_c is minimum for all $c \in C$. Thus, the maximum separability index S_I^{max} is given by

$$S_I^{max} = \frac{1}{l} \sum_{c \in C} S_c^{max} = \frac{1}{l} \sum_{c \in C} 1 = \frac{l}{l} = 1, \tag{24}$$

and the minimum separability index S_I^{min} is given by

$$S_I^{min} = \frac{1}{l} \sum_{c \in C} S_c^{min} = \frac{1}{l} \sum_{c \in C} 1/l = \frac{l/l}{l} = \frac{1}{l}. \tag{25}$$

□

Algorithm 4 illustrates and summarizes all steps for computing the class separability index of a given dataset. Besides all the steps above, in Line 5, we have to perform an additional step to consider the CCEP limits. This step is necessary since the KDE method only estimates the density with a grid according to the parameters and is not aware of these limits. However, since they are strict for the CCEP, assuring all points are always within limits, we have to change all grid elements outside the CCEP limits to zero. Otherwise, it will impact the metrics computation. It is

worth noting that the robust clustering Skinny-Dip presents a significance level and is used as a threshold for estimating the points within their clusters, and, thus, we set $\alpha = 0.05$.

The class separability index S_I depends on a given dataset and other instances may show different S_I values. This fact may cause differences, for instance, between the expected classification potential when analyzing a given training dataset, and the resulting accuracy for the testing dataset.

ALGORITHM 4: ClassSeparability

Input: Lists of computed features from the m time series of a given dataset, $\mathbf{h} = \{h_1, \ldots, h_m\}$ and $\mathbf{s} = \{s_1, \ldots, s_m\}$, for the normalized Shannon entropy and statistical complexity, respectively. Their labels $\mathbf{y} = \{y_1, \ldots, y_m\}$, where $\forall i,\ y_i \in C = \{c_1, \ldots, c_l\}$. And an embedding dimension D.

Output: The class separability index S_I for the given parameters.

```
// Estimating density grids for each class
for k ← 1 to l do
    // Filtering features by class
    h ← {h_j : y_j = c_k} ; s ← {s_j : s_j = c_k};
    // Filtering outliers features with skinny-dip
    h, s ← skinnyDipCluster (h, s, α ← 0.05);
    // Computing the grid of class k with KDE
    A^k ← KDE (h, s, n_KDE ← 250);
    // Zeroing grid elements outside the CCEP limits
    A^k ← limitCCEP (A^k);
    // Converting densities to probability
    a^k_{i,j} ← a^k_{i,j} / Σ_{i=1}^{n_KDE} Σ_{j=1}^{n_KDE} a^k_{i,j}
// Normalizing each element of the classes as [0,1]
for i ← 1 to n_KDE do
    for j ← 1 to n_KDE do
        for k ← 1 to l do
            a^k_{i,j} ← a^k_{i,j} / Σ_{c∈C} c_{i,j}
// Computing individual separabilities for each class
for k ← 1 to l do
    // The number of elements in the k-th grid with non-zero values
    A' = {a_{i,j} | a_{i,j} > 0}| ; m ← |A'|;
    // Computing the individual separability for the class
    s_k ← Σ_i Σ_j a'_{i,j} / m
// Computing the class separability index
S_I ← 1/l Σ_{c∈C} s_c
return S_I;
```

5 TSCLAS: TIME SERIES CLASSIFICATION VIA CLASS SEPARABILITY ANALYSIS

The class separability index S_I measures the potential for correctly distinguishing time series according to their dynamics, and, thus, it is reasonable to consider the maximization of S_I to achieve better classification results. For the domain of ordinal patterns transformations, both embedding dimension D and embedding delay τ have important roles in capturing different aspects of time series dynamics, which could indicate such maximization task [22, 42, 44, 61].

Concerning the embedding dimension D, as emphasized in Section 4, there are recommendations for its proper definition, which is directly related to the time series length. Since $D!$ is the number of possible ordinal patterns (symbols) to consider, the length n of time series must be long

enough so that the sampling in the $D!$ space of patterns is representative, i.e., $n \gg D!$ [44]. On the other hand, there is no simple rule of thumb for defining the embedding delay τ. Although for most studies, it is sufficient to define $\tau = 1$, different values of τ are used for specific tasks, such as finding periodicity in data and estimating the scale of delayed systems [61]. Furthermore, as shown by Zunino et al. [61] and Borges et al. [7], it is possible to discover important characteristics of the time series when evaluating the extracted metrics as a function of τ.

Thus, in this work, we follow the approach of analyzing the behavior of the classes' metrics for different values of embedding delays τ, intending to find the most appropriate one for the maximization of S_I. This is formally defined by Problem 5.1.

Problem 5.1. *For a given dataset $\mathcal{D}$ and an embedding dimension D, find $\tau \in \mathcal{T}$ that maximizes the class separability index S_I. The search space $\mathcal{T} = \{1, \ldots, \tau_{max}\}$ consists of the set of possible embedding delays, with respect to both D and the length n of time series in $\mathcal{D}$.*

5.1 Maximization of S_I Via Multiscale Approach

The maximization of class separability index via a multiscale approach consists of computing S_I for each pair (D, τ), with $\tau = \{1, \ldots, \tau_{max}\}$. Different values for τ correspond to sampling data with different interval scales within the time series data points. In this case, we have a limit for the maximum possible value of τ_{max}, which depends on the length n of the time series and the chosen embedding dimension D, and is bounded by

$$\tau_{max} < \frac{n}{D-1}. \tag{26}$$

This limit affects the number of observed ordinal patterns within a time series, so this bound ensures at least one observed pattern. For the experiments from this work, we have datasets with time series ranging from $n = 96$ to $n = 30240$, representing data collected at 1-minute intervals for 1 day up to 3 weeks. If we consider $D = 6$, this range gives a $\tau_{max} = 19$ for the former case and $\tau_{max} = 6{,}048$ for the latter. However, such large values of τ are unfeasible in practice, since a higher τ will create sliding windows with very distant elements from the time series, losing any relation among them. For our analysis, we have decided to set $\tau_{max} = \min(\tau_{max}, 30)$, which we empirically found as a reasonable tradeoff between capturing the temporal correlation dependence of the time series and taking into account the computational costs of the algorithm.

Algorithm 5 presents our strategy for finding the best τ^* that maximizes the separability index S_I, for a given dataset. The method consists of computing the S_I for each τ within the list of embedding delays $\mathcal{T}$, Lines 2–11. For a given pair (D, τ), the normalized Shannon entropy and statistical complexity are computed from the ordinal patterns probability distributions of each time series (Lines 5–8). With these features, we can create the CCEP and compute its class separability index (Line 9). After these rounds, the maximum τ is chosen (Lines 10 and 11).

5.2 Time Series Classification Strategy

TSCLAS is based on a combination of features extracted from the ordinal patterns probability distribution (p_π) and the ordinal patterns transition graph (G_π), both obtained from the set of ordinal patterns (Π) from these time series. To compute those features, we consider the ordinal patterns transformations for a given embedding dimension D and, for the embedding delay, we chose the best τ^* according to the strategy presented in Algorithm 5, which is the one that maximizes the class separability index S_I for the present data.

The whole classification strategy is presented in Algorithm 6. This algorithm assumes as inputs a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, which is composed by m time series $\mathbf{x}_i = \{x_1, \ldots, x_n\}$ of size n each, with each of them having its label in $\mathbf{y} = \{y_1, \ldots, y_m\}$. As parameters for the ordinal patterns

ALGORITHM 5: FindTau

Input: A dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, containing m time series $\mathbf{x}_i = \{x_1, \ldots, x_n\}$ of size n each, with labels $\mathbf{y} = \{y_1, \ldots, y_m\}$. An embedding dimension D, and a list of embedding delays $\mathcal{T} = \{\tau_1, \ldots, \tau_l\}$.
Output: The τ^* that maximizes the separability index S_I of $\mathcal{D}$.

```
1  S_I ← 0;
2  forall the τ ∈ T do
       // Lists for the features extracted from each of the m time series of D
3      h ← {h_i = 0 : i = 1, ..., m};
4      s ← {c_i = 0 : i = 1, ..., m};
5      for i ← 1 to m do
           // Computing the ordinal patterns probability distributions for the pair (D, τ)
6          p_π ← OrdinalPatternsPD(x_i, D, τ);
           // Features extracted: normalized Shannon entropy and statistical complexity
7          h_i ← ShannonEntropy(p_π);
8          s_i ← StatisticalComplexity(p_π);
       // Calculating the class separability index
9      S'_I ← ClassSeparability(h, s, y, D);
10     if S'_I > S_I then
11         τ* ← τ
12 return τ*;
```

ALGORITHM 6: Classification

Input: A dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, containing m time series $\mathbf{x}_i = \{x_1, \ldots, x_n\}$ of size n each, with labels $\mathbf{y} = \{y_1, \ldots, y_m\}$. An embedding dimension D, and a list of embedding delays $\mathcal{T} = \{\tau_1, \ldots, \tau_l\}$.
Output: The predicted labels $\mathbf{y}_{pred}$ for the test dataset.

```
   // Dataset train/test split
1  D_train, D_test ← TrainTestSplit (D, train_pct);
   // Finding the best tau for the train set
2  τ* ← FindTau (D_train, D, T);
   // Computing features for the selected tau
3  F_train ← ExtractFeatures (D_train, D, τ*);
4  F_test ← ExtractFeatures (D_test, D, τ*);
   // Pre-procesing the features: center and scale
5  F_train ← ScaleData (F_train);
6  F_test ← ScaleData (F_test);
   // Training step
7  model ← Training (F_train)
   // Prediction step on test datset
8  y_pred ← Prediction (model, F_test)
9  return y_pred;
```

transformations, it expects an embedding dimension $D \in \mathbb{N}$, and a list of embedding delays $\mathcal{T} = \{\tau_1, \ldots, \tau_l\}$ to look for the best τ^*.

After randomly splitting the datasets into the training and testing subsets, $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$, respectively, in Line 1, the next step is to obtain the τ^* that maximizes the class separability for the training split, Line 2. It is worth mentioning that, following this method for selecting τ^*, we avoid the need to perform successive classifications on a validation set as in a conventional hyper-parameter tuning approach. Instead, we only need to evaluate the maximization of the proposed separability index with the training set, which is more efficient and adequate for the IoT scenarios.

Thus, once we have D and τ^*, we can extract the features from both datasets (Lines 3 and 4), which are further scaled to have zero mean and unit variance (Lines 5 and 6). The computed features are those described in Section 3.4 and presented here as $\mathcal{F}_{train}$ and $\mathcal{F}_{test}$ from the training and testing subsets, respectively.

For the time series classification step, once we have the dataset properly converted, any classification algorithm may be used, according to the availability and necessity of the problem. For the chosen classifier, the training and prediction steps are presented in Lines 7 and 8, respectively. However, as shown in Section 6, given the importance of considering the large numbers of IoT environments, we chose the random forest classifier [10], so this step may correspond to both accuracy and processing time requirements. Section 6.1 presents more details regarding the chosen classification algorithm and its parameters.

6 RESULTS

To evaluate TSCLAS, we performed experiments of time series classification on datasets collected from real-world sensors, and compared our results with known classification algorithms from the literature. Besides, we considered common data quality issues to evaluate the robustness of TSCLAS. In the following, we describe the materials and methods for our experiments, and the results achieved for the considered scenarios.

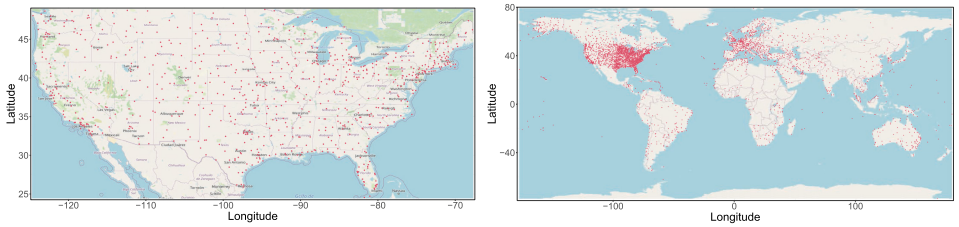
6.1 Materials and Methods

As described by Borges et al. [8], Borges Neto et al. [9] and Montori et al. [33], IoT data is very challenging to handle. Besides the large amount and heterogeneity of sensors, issues such as missing readings, different rates, among others, make IoT a very unreliable scenario. Thus, in order to properly evaluate TSCLAS, we have to choose those sensors from the whole IoT spectrum that can provide more reliable data, mainly with respect to their precision, probability of correctness, and trustworthiness [9, 11]. We decided to perform our experiments using data from meteorological sensors at airports, the **Automated Surface Observing Systems (ASOS)**. These are reliable sensors that generate observations every minute, or every hour, according to the airport, and are used to support weather forecast activities and aviation operations.[2] Their reliability comes from constant data quality monitoring, 24 hours per day, with maintenance as soon as a problem is detected.

The typical format of these data is the METAR format,[3] standardized by the **International Civil Aviation Organization (ICAO)**, which follows a common syntax used for weather reporting around the world.[4] However, to collect our data, we use the weather data archive of the **Iowa Environmental Mesonet (IEM)**, from the Iowa State University, which collects the ASOS data, stores and makes them available via an API and a Web interface.[5] In our experiments, we collect data with a 1-minute interval, available only for the ASOS from the United States, and with a 1-hour interval, available for ASOS from several countries in the world.

With these data, we can perform experiments with different setups by varying the data interval granularity, time span, and geographical locations. Thus, we evaluate TSCLAS on time series of (i) temperature, (ii) atmospheric pressure, (iii) wind speed, and (iv) wind direction, composing the four classes in our datasets, considering both 1-minute and 1-hour time intervals. For the

[2] Automated Surface Observing Systems from the U.S. National Weather Service: https://www.weather.gov/asos/asostech.
[3] METAR description at Wikipedia: https://en.wikipedia.org/wiki/METAR.
[4] Guide to decode the ASOS - METAR format: https://www.weather.gov/media/wrh/mesowest/metar_decode_key.pdf, and a more intuitive METAR decoder (in portuguese): https://www.redemet.aer.mil.br/?i=facilidades&p=decodificacao-metar.
[5] Iowa Environmental Mesonet ASOS-METAR Data Download: https://mesonet.agron.iastate.edu/ASOS/.



(a) ASOS locations of 803 airports in the United States used for the 1-minute dataset.

(b) ASOS locations of 3742 airports from around the world used for the 1-hour dataset.

Fig. 2. Geographical locations of the airports where ASOS data were collected.

1-minute interval experiment, the time series data were collected from 803 airports in the U.S. with geographical locations illustrated in Figure 2(a), and, for the 1-hour interval, the time series data were collected from 3,742 airports around the world, as depicted in Figure 2(b). The validity of our experiments is corroborated by the fact that the time series related to weather phenomena are the most common types of data in the current IoT [8, 9, 33].

We may choose any classification algorithm to perform the time series classifications, as described in Algorithm 6. However, given the characteristics of IoT scenarios, we decided for the random forest classifier [10]. Since we are dealing with large IoT datasets, this decision is motivated by the good reported accuracy and fast processing time of the random forest classifier, besides its robustness to outliers and noise. Furthermore, given the relevance of the computed features for each time series, the importance of the random forest internal estimation variable is a proper solution for the classification. For the random forest classifier, two main parameters must be given: the number of trees in the forest, where we fixed as $ntree = 200$, and the number of variables randomly sampled as candidates at each split, also fixed as $mtry = 2$ in our experiments. The decision to fix these parameters and not perform any parameter optimizations is to conduct a fair comparison between all algorithms, independently of their programming languages and libraries specificities.

In order to evaluate our proposal with respect to the related work, we decided to compare our results with known classification algorithms from the literature. As described in Section 2, the strategies that are specifically designed for the classification of IoT data are very domain specific, making it difficult to adapt to other contexts, or the authors of those articles does not make their code available, making them difficult to be reproducible. Thus, TSCLAS is compared to four different classification algorithms from literature, which are well known and easy to reproduce. The **Random Forest classifier (RANDF)**, to test the classification of the raw time series without our transformation step; the **k-nearest neighbors (KNN)**, to compare with this well-known distance-based algorithm, mostly used as a ground truth for time series classification with $k = 1$, and the **Dynamic Time Warping (DTW)** as a distance method [3, 38]; the **Time Series Forest Classifier (TSF)** [14], a composition algorithm that improves the random forest method; and the **Random Interval Spectral Forest (RISE)** [24], a more recent frequency-based algorithm that performs an ensemble of the trees with features extracted from the spectral domain of time series. For these tree-based algorithms, we also fix the number of trees as $n\_estimators = 200$ and the number of splits as $min\_samples\_split = 2$, to have a fair comparison between all methods and our strategy.

All experiments were performed in a computer with Intel Core i9-9900X CPU at 3.50 GHz × 20, 128 GB RAM, and running a Linux Ubuntu 18.04.4 LTS 64-bit. We implemented TSCLAS in R 3.4.4 [52], with some excerpts of code in C++. All pieces of code used in our experiments are

Table 1. List of the Main Software and Libraries Versions Used in the Experiments in This Work

Software	Details	Version
R	R software system.	3.4.4
caret	R package used for preprocessing data (centering and scaling) and for creating the dataset splits.	6.0.86
randomForest	R package used for the RANDF classifier.	4.6.14
diptest	Hartigans' Dip Test for Unimodality, used for the skinny-dip clustering.	0.75-7
MASS	Used for the kernel density estimation KDE.	7.3-49
Rcpp	Used for the codes excerpts in C++.	1.0.4.6
matrixcalc	Used for matrix operations when extracting features.	1.0-3
Python	Python software system.	3.6.9
scikit-learn	A machine learning library for the Python. Used for the RANDF classification algorithm and accessory functions.	0.23.1
pandas	Python tool used for data analysis and manipulation tool.	1.1.0
numpy	A library for the Python programming language used for mathematical operations.	1.19.1
sktime	A unified toolbox for machine learning with time series. Used for the time series classification algorithms of literature.	0.4.1

available at a public repository.[6] The algorithms from the literature used to compare are implemented in Python 3.6.9, within the *sktime* library [27], a unified toolbox for machine learning with time series, except for the RANDF, which is implemented with the scikit-learn library [35].

Table 1 presents the versions of software and libraries used in our experiments.

Another step necessary when comparing TSCLAS to the literature solutions is to transform the raw collected time series into a proper dataset to be read by the algorithms. Although not necessary to TSCLAS, for the other strategies we have to first adjust the time series to have equal length. For this, given the expected number of data points for a given time span and a fixed time interval between the points, we adjusted each observation from the collected data by its timestamp to its corresponding position in the adjusted time series. For instance, for making a dataset from the 1-minute raw time series data, considering a 1-day time span, with a 1-minute interval between observations, each time series should have 1,440 samples. Thus, by considering a starting date, which for our experiments was January 1, 2020, we place each observation at the correct position according to its timestamp. For timestamps without observations, we have a missing data sample. For evaluating TSCLAS in different setups, we create different datasets by varying these parameters, which impact the time series lengths for each case.

For the 1-hour raw time series, the time interval is fixed at 1 hour, but we may vary the time span. However, for these time series, before adjusting the timestamp, we have to synchronize the hourly observations from different time series according to their minute of data sampling. This is necessary because each ASOS sensor collects data at different minutes from the hour. For instance, some of them make their samples at 00:05h, 01:05h, 02:05h, and so on, while others collect their samples at 00:42h, 01:42h, 02:42h, and so on. Thus, we discover each minute of collection for the ASOS sensors by considering the minute where most of the data is present. After that, we consider this minute as the offset for this sensor and use it to center all data samples at minute 0 of each hour. Although this may potentially distort the moment of data collection by at most 59 minutes, it is necessary to create the dataset for the literature algorithms. It is important to mention that this issue does not impact the ordinal patterns construction of our strategy.

For both 1-minute and 1-hour datasets, we only consider adding to the dataset those time series with at least 80% of valid data samples for the given time parameters. For those time series satisfying this condition, before classifying them using literature algorithms, we apply a linear interpolation for missing data imputation. We have tested different approaches for the data

[6]Classification of IoT data with ordinal pattern transformations: https://github.com/labepi/tsclas. This code requires the codes for Bandt-Pompe ordinal pattern transformations, available at https://github.com/labepi/bandt_pompe.

Table 2. Configurations for the 1-minute and 1-hour Experiments and the Lengths of Time Series for Each Time Span and Interval Parameters

1-minute experiments					1-hour experiments			
Time span	Time series length per interval				Time span	Time series length	Time span	Time series length
	1 min	5 min	10 min	15 min				
1-day	1440	288	144	96	1-month	744	4-months	2904
1-week	10080	2016	1008	672	2-months	1440	5-months	3648
2-weeks	20160	4032	2016	1344	3-months	2184	6-months	4368

imputation, and our conclusion is that a simple linear approach provides the best trade-off between the resulting accuracy and processing time.

Table 2 presents the different configuration setups used in our experiments.

Finally, another parameter necessary for the classification step is the training percentage used for the dataset train/test split, which is defined as $train_{pct} = 0.8$ for all experiments involving dataset split, giving the 80/20 split as commonly used in the machine learning literature. We also assure that each one of the four classes present in our data has the same number of instances on both train and test splits, so the datasets are balanced with respect to the number of classes. Finally, we assure the precision of our results by presenting them as the average of 30 resamples of randomly selected train/test splits, with 95% confidence interval of the error margin.

6.2 Discussion

In this section, we present our results on the classification of time series for the 1-minute and 1-hour ASOS time series. To evaluate TSCLAS and compare its behavior in contrast to the known solutions from literature, we perform three different experiments, as presented in the following.

6.2.1 Classification of 1-Minute Time Series. In this experiment, we evaluate the capacity of TSCLAS to correctly classify the 1-minute ASOS time series under different conditions, varying time span of data collection, and time interval between consecutive observations. Figure 3(a) presents the classification accuracy of TSCLAS by considering four different embedding dimensions $D = \{3, 4, 5, 6\}$, and for the KNN, RANDF, TSF, and RISE, algorithms for the 1-day time span. Table 3 presents all mean accuracies for these regular time series, best results per configuration are in bold.

We can observe that, for the 1-minute time interval, TSCLAS achieves an accuracy around 0.89, behind TSF and RISE, which are in the order of 0.91, and ahead of KNN and RANDF. We can also see that our accuracies considerably decay as the time interval between samples increases for 5, 10, and 15 minutes, being more accentuated for higher values of D. While KNN and RANDF remain barely constant, similar behavior is also observed for both TSF and RISE, but more smoothly. This expected behavior was presented in Section 3.1, i.e., for a given time series with length n, the condition $n \gg D!$ must be satisfied to obtain reliable statistics with the ordinal patterns transformations [61]. Since we only have 96 samples for 15-minute time intervals in a single day, c.f. Table 2, there is not enough data for the method, mainly when D is higher.

On the other hand, as presented in Figure 3(b), as the number of samples increases when considering 1 week of data, the achieved accuracies also increase. For instance, for a 1-minute time interval, the time series length grows from 1,440 to 10,080 observations, and we have accuracies around 0.94 for all values of D, tying whith TSF. However, we can also observe the dependency of TSCLAS on the number of samples, as the time interval increases and, consequently, the n decreases up to 672 samples for 15 minutes, decreasing our achieved accuracies.

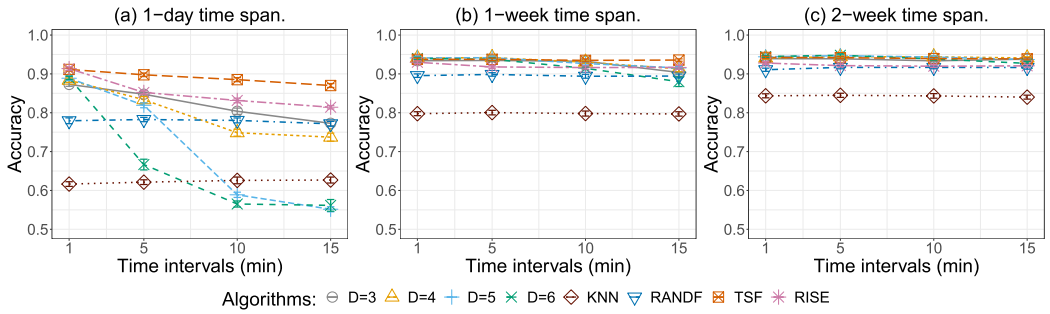


Fig. 3. Accuracies of 1-minute ASOS time series for time spans of (a) 1-day, (b) 1-week, and (c) 2-week, starting in January 1, 2020, evaluated for time intervals of 1, 5, 10, and 15 minutes between consecutive observations.

Table 3. Classification Accuracies for Different Configurations of the 1-minute Experiment, for the Regular Time Series and their Versions with Missing Data Gaps

Time span	Time interval	D=3	D=4	D=5	D=6	KNN	RANDF	TSF	RISE	D=3	D=4	D=5	D=6	KNN	RANDF	TSF	RISE
		Regular time series								**10 % gap**							
1-day	1 min	0.87	0.88	0.89	0.89	0.62	0.78	**0.91**	**0.91**	0.87	0.87	0.89	0.88	0.61	0.77	0.90	**0.91**
	5 min	0.85	0.83	0.82	0.67	0.62	0.78	**0.90**	0.85	0.83	0.83	0.80	0.71	0.62	0.77	**0.87**	0.84
	10 min	0.80	0.75	0.59	0.57	0.63	0.78	**0.89**	0.83	0.79	0.75	0.75	0.54	0.62	0.77	**0.86**	0.82
	15 min	0.77	0.74	0.55	0.56	0.63	0.77	**0.87**	0.81	0.75	0.72	0.70	0.52	0.62	0.77	**0.86**	0.80
1-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.80	0.90	**0.94**	0.93	0.93	**0.94**	**0.94**	**0.94**	0.79	0.89	0.93	0.93
	5 min	0.93	**0.94**	**0.94**	**0.94**	0.80	0.90	**0.94**	0.92	0.93	**0.94**	**0.94**	0.93	0.78	0.88	0.93	0.91
	10 min	**0.93**	**0.93**	**0.93**	0.91	0.80	0.89	**0.93**	0.92	0.92	**0.93**	0.92	0.91	0.78	0.88	**0.93**	0.91
	15 min	0.90	0.91	0.92	0.88	0.80	0.89	**0.94**	0.92	0.91	0.90	0.91	0.89	0.79	0.88	**0.92**	0.91
2-week	1 min	**0.94**	**0.94**	**0.94**	**0.94**	0.84	0.91	**0.94**	0.93	**0.94**	**0.94**	**0.94**	**0.94**	0.82	0.89	**0.94**	0.93
	5 min	0.94	0.94	**0.95**	**0.95**	0.84	0.92	0.94	0.92	**0.94**	**0.94**	**0.94**	**0.94**	0.83	0.90	0.93	0.92
	10 min	0.93	**0.94**	**0.94**	**0.94**	0.84	0.92	**0.94**	0.92	0.93	**0.94**	**0.94**	0.93	0.82	0.90	0.93	0.92
	15 min	**0.94**	**0.94**	**0.94**	0.93	0.84	0.92	**0.94**	0.92	**0.94**	**0.94**	0.93	0.93	0.82	0.90	0.93	0.92
		30% gap								**50 % gap**							
1-day	1 min	0.84	0.86	0.85	0.85	0.58	0.72	0.88	**0.89**	0.82	0.82	0.81	0.77	0.56	0.68	**0.85**	0.81
	5 min	0.79	0.76	0.75	0.73	0.58	0.73	**0.84**	0.81	0.74	0.72	0.70	0.67	0.55	0.68	**0.81**	0.73
	10 min	0.74	0.69	0.70	0.66	0.60	0.73	**0.82**	0.79	0.66	0.62	0.59	0.53	0.56	0.67	**0.79**	0.72
	15 min	0.68	0.68	0.65	0.59	0.60	0.72	**0.81**	0.77	0.60	0.58	0.51	0.51	0.57	0.67	**0.76**	0.69
1-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.71	0.82	0.92	0.92	0.92	**0.93**	**0.93**	**0.93**	0.70	0.79	0.91	0.87
	5 min	0.92	**0.93**	**0.93**	**0.93**	0.72	0.83	0.91	0.89	**0.92**	**0.92**	**0.92**	0.91	0.70	0.79	0.89	0.81
	10 min	**0.92**	**0.92**	0.91	0.89	0.72	0.82	0.90	0.89	**0.91**	0.90	0.89	0.87	0.70	0.79	0.88	0.82
	15 min	0.90	**0.91**	0.88	0.87	0.73	0.82	0.89	0.88	**0.89**	0.86	0.84	0.83	0.69	0.79	0.87	0.80
2-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.76	0.85	0.92	0.92	0.93	**0.94**	**0.94**	**0.94**	0.72	0.82	0.92	0.90
	5 min	**0.94**	**0.94**	**0.94**	**0.94**	0.76	0.85	0.92	0.91	0.93	**0.94**	**0.94**	0.93	0.72	0.82	0.91	0.88
	10 min	**0.94**	**0.94**	**0.94**	0.93	0.76	0.85	0.91	0.90	**0.93**	**0.93**	**0.93**	0.92	0.72	0.81	0.90	0.87
	15 min	0.92	**0.93**	**0.93**	0.91	0.75	0.86	0.91	0.90	**0.93**	0.92	0.91	0.90	0.72	0.82	0.90	0.87

Best results per configuration are highlighted in bold.

The length of the time series is an important aspect to consider, but it is not the only one impacting our classification results. For instance, the length of the time series for the 1-week experiment with a 15-minute interval of Figure 3(b) is shorter than the time series for the 1-day experiment with a 1-minute interval of Figure 3(a). However, the accuracy of the former is higher than the

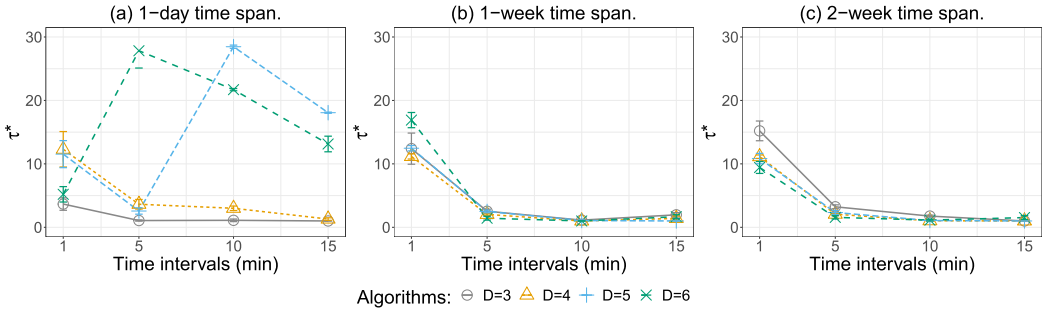


Fig. 4. Selected values of τ^* during the maximization of class separability index for the configurations of 1-minute ASOS time series, for (a) 1-day, (b) 1-week, and (c) 2-week time spans.

latter, i.e., around 0.9 for all Ds, except for $D = 6$, which has 0.88, the same as before. This behavior occurs because the method's success depends on two aspects: number of observations, as discussed above, and representativeness of the phenomena of the collected data.

For instance, the temperature values of METAR are reported as degree Celsius with two integer digits, without decimal places, which are rounded up. The same rounding is also applied to other data types. Thus, with an 1-minute interval, small temperature variations are lost, and the values remain constant as long as the temperature does not increase/decrease enough. These constant numbers are misinterpreted by the method as a deterministic behavior that does not exist. However, by increasing the time interval between data observations, the number of consecutive samples tends to reduce, and the phenomena' real conditions can be correctly represented by data. For the 1-minute interval samples, the increase in the number of observations masquerade some data representation issues, and we have the best accuracy results for that case. The same behavior is observed for a 2-week time span, presented in Figure 3(c), but with better results for all time intervals, once the number of samples for 15 minutes is large enough. In general, TSCLAS ties with the best classifier, TSF, even beating it in the configuration of 2-week and 5-minute interval.

The issue of the time series length versus the representativeness duality is also present in TSCLAS during the maximization of the class separability index. Figure 4 presents the average selected τ^* for 1-minute configurations. We can see that, for the 1-week and 2-week spans, in Figures 4(b) and 4(c), respectively, as the time interval increases the selected τ tends to 1, since the spacing between observations already provides a significant data representation. On the other hand, for 1-minute intervals, we need large values of τ to minimize the impact of rounding.

For the 1-day case, presented in Figure 4(a), since we have small time series, this behavior is observed only for $D = 3$ and $D = 4$. For $D = 5$ and $D = 6$, we have an inconsistent behavior of the selected τ's because the length n of time series does not satisfy the recommendation for $n \gg D!$, as previously discussed in Section 3.1 [42, 44]. This reinforces the fact that larger values of D are not recommended for the configurations with short time series since the method cannot compute enough patterns to represent the phenomenon.

Concerning the time spent on the time series classification, Figure 5 presents the average times for the training and prediction steps for the 1-minute ASOS time series. Overall, the time spent for both training and testing increases with the data length. The algorithms with lower processing times are KNN and RANDF, followed by TSCLAS, and, finally, TSF and RISE. For TSCLAS, the time increases with the embedding dimension D, which can easily be observed in Figures 5(b) and 5(c) of the training steps, but it is also similar to the other configurations. TSCLAS requires more training time than KNN and RANDF, but presents better classification accuracies. Given the current

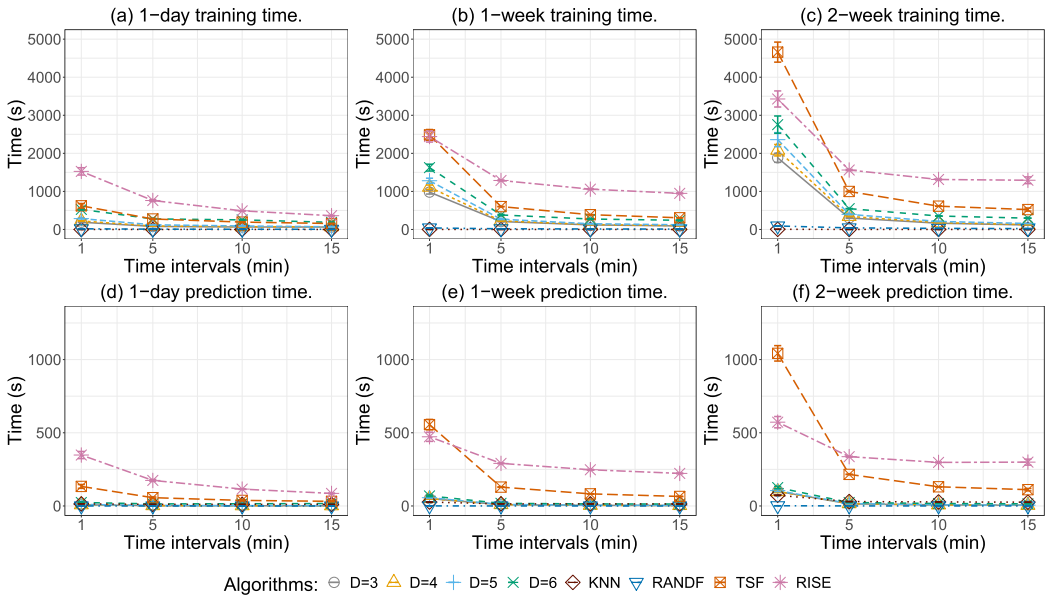


Fig. 5. Average time spent for the classification of 1-minute ASOS time series, considering their training steps with (a) 1-day, (b) 1-week, and (c) 2-week times spans, and their prediction steps for (d) 1-day, (e) 1-week, and (f) 2-week time spans.

availability of cloud and fog resources, the training step can be processed in those powerful machines, which might favor our strategy. Besides, the time spent in the prediction step is competitive to KNN and RANDF, as can be observed in Figures 5(d)–5(f).

In summary, when TSCLAS is applied to long-enough time series, for configurations of 1 week and 2 weeks, it beats the accuracies of KNN and RANDF and is very close to TSF and RISE, even beating them for some configurations. Furthermore, even when not winning these last two, its processing times for both training and prediction are lower, making it a competitive strategy. On the contrary, for the 1-day configurations, TSCLAS is not the best solution since the conditions are not appropriate to achieve its full potential.

Another perspective for evaluating TSCLAS is presented in Figure 6, which illustrates our worst- and best-case scenarios for each embedding dimension D. Our worst results occur when there are fewer observations per time series, thus, Figures 6(a) and 6(b) illustrate the confusion matrices for the experiments configured with 1-day time span and 15 minutes of time interval between observations, for $D = \{3, 4, 5, 6\}$, respectively. Each confusion matrix was obtained by accumulating the actual-predicted classes for all 30 resamples from each experiment configuration, which was then normalized between 0 and 1 and presented as heat maps. From those figures, we can observe the errors increasing as a function of D, but those errors occur in some particular ways. For instance, it is more likely to misclassify temperature with atmospheric pressure and vice-versa. The same occurs between the wind direction and wind speed time series, which are more misclassified. This behavior occurs due to the similarity between these phenomena' intrinsic dynamics, which are easier to confuse when there is not enough data.

Concerning the best classification results, Figures 6(e)–6(h), present the heat maps of the confusion matrices for the experiments with a 2-week time span and 1-minute time interval between observations, which are the configurations with more data for the considered embedding

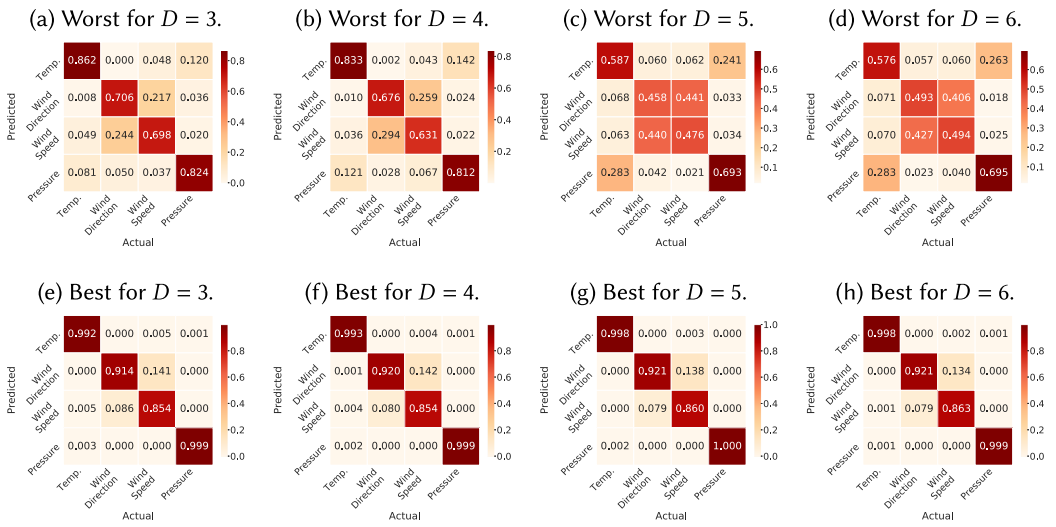


Fig. 6. Heat maps of the confusion matrices for the worst and best results, with respect to the classification accuracies, for $D = \{3, 4, 5, 6\}$. Worst results occur for 1-day time span and 15 minutes of time interval between observations, and the best ones are achieved for 2-week time span with 1 minute of time interval.

dimensions. We can note the reduction of classification errors for all values of D, but there is still a concentration of errors among the wind direction and wind speed time series. Since both time series represent distinct views of the same physical phenomenon (wind), it is harder to make a perfect distinction among them. This issue is inherited from the ordinal pattern transformations, which go deep into the phenomena' dynamic characteristics. At this point, the time series from temperature and atmospheric pressure are correctly classified 99% of the time.

6.2.2 Classification of 1-Minute Time Series with Missing Data. To evaluate the robustness of TSCLAS, we present the results for experiments when we have missing data in the time series. We simulated the case where sensors stopped generating data for a while, and then return their operation, creating continuous gaps in time series. These are concerns that easily occur in current IoT scenarios and that we must consider when classifying their data [8, 9]. The problem with gaps is that (i) they reduce the number of observations, which may affect our method as previously mentioned, and (ii) it is harder to perform data imputation as the gaps increases, which may affect the performance of other methods.

To evaluate at what extent this is a problem for the compared strategies, we tried three different gap sizes, which are 10%, 30%, and 50% of the original length of time series. For each time series, given a gap size, the moment in time the gap occurs is randomly selected within the time series. Starting from the point corresponding to this selected moment in time, all consecutive data points of the time series are removed until completing the gap length for each case. The achieved accuracies for the algorithms for the 1-minute datasets, considering different time spans, time intervals, and gap sizes. are illustrated in Figure 7 and its numerical results are presented in Table 3.

Figures 7(a)–7(c) present the impacts of the increase in the gap size in the algorithms' accuracy for the 1-day time span. For TSCLAS, we can observe this impact by the steeper negative slope as the gap size increases. In contrast to Figure 3, which consists of the classification of the original data without gaps, by reducing the number of observations with gaps, the accuracy of TSCLAS is more damaged. This is due to the violation of the $n \gg D!$ requirement for the transformation,

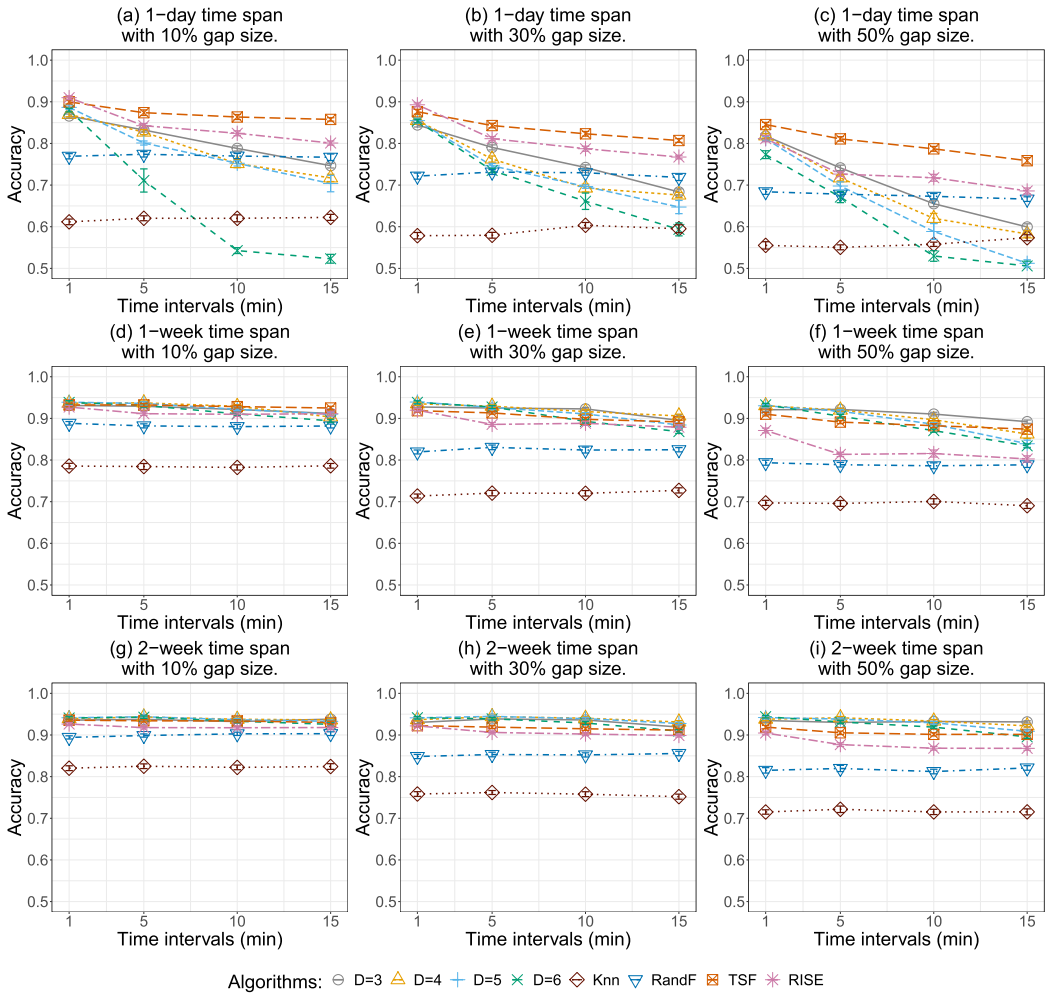


Fig. 7. Classification accuracies of 1-minute ASOS time series with missing data, simulated as continuous gaps randomly positioned within the time series. Figures correspond to combinations 1-day, 1-week, and 2-week time spans, for gaps with 10%, 30%, and 50% of the length of the time series, evaluated for different time intervals of 1, 5, 10, and 15 minutes between consecutive observations.

as explained in Section 3.1. For instance, for a 1-day time span, with a 15-minute interval, and 50% of gaps in data, we only have 48 observations to compute the ordinal patterns. The impact on other strategies was only noted for 30% and 50% of gaps. While TSF and RISE have a similar but smoother effect than ours, RANDF and KNN have their general accuracy lower but constant concerning the increase in the time intervals.

This same constant, but lower, behavior of RANDF and KNN also occurs for these algorithms on the 1-week and 2-week time spans, presented in Figures 7(d)–7(f) and 7(g)–7(i), respectively. For other strategies, the increase in gaps still impacts their accuracy, but this is mitigated as the time span increases. However, as the number of observations per time series increases with the growth in time span, the impact of gaps is less perceived on our strategy than on the TSF and RISE. For instance, considering Figure 7(i) and Table 3, for 2-week time span with a 50% gap size,

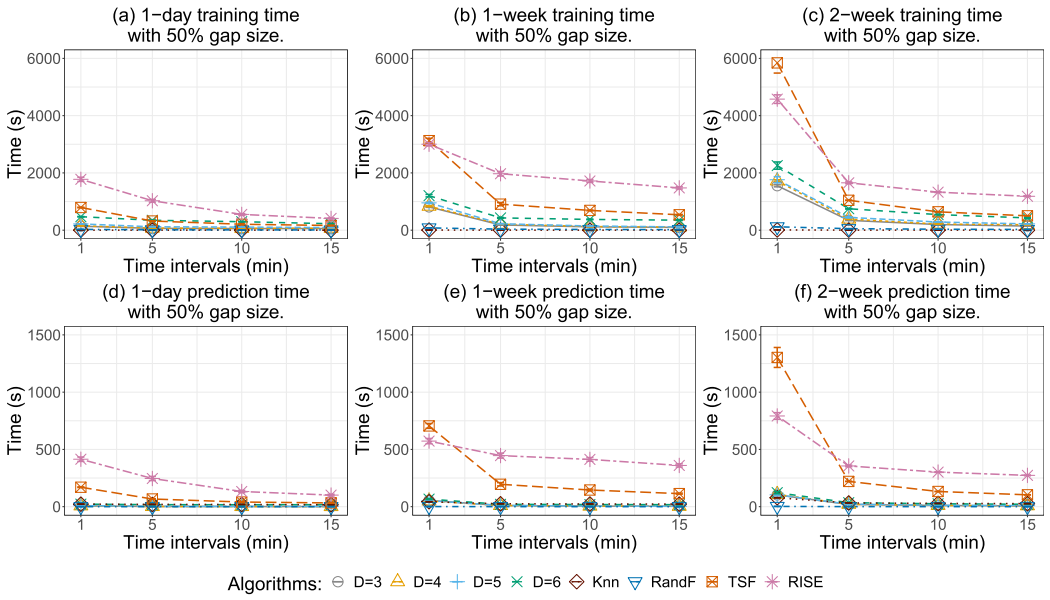


Fig. 8. Average time spent for the classification of 1-minute ASOS time series with 50% of time series samples as missing data. The missing data points were simulated as continuous gaps, randomly positioned within the time series. We consider evaluating the training step with (a) 1-day, (b) 1-week, and (c) 2-week times spans, and the prediction step for (d) 1-day, (e) 1-week, and (f) 2-week time spans.

the best accuracies for TSCLAS, for different configurations and values of D, range from 0.93 to 0.94, beating the best results of other compared classifiers. This behavior represents a minimal impact of gaps in TSCLAS, reinforcing our claim that, once we have the representativeness of the phenomena on the collected data, our performance is a matter of enough observations in the time series.

We evaluate the time spent by the methods to classify the time series with missing data. Without loss of generality, we take as an example the case for 50% gaps for both training and prediction steps, presented in Figure 8. By comparing these times to the ones for the classification steps without gaps, presented in Figure 5, we can see similar behavior for all methods. However, while TSCLAS has a reduction in the time spent for both training and prediction steps in general, the other methods increased their spending times. This occurs due to the reduced time series length as the gap increases, which, for our method, also reduces the number of ordinal patterns to compute.

The methods that require some data imputation need more pre-processing time to adjust the data, increasing their total time. Take TSF, for instance, the most prominent case for a 2-week time span with 1-minute time interval. It increased its training and prediction times from approximately 4,700 and 1,040 seconds, to approximately 5,840 and 1,300 seconds, respectively. Oppositely, our worst-case scenario, with $D = 6$, decreased its training and prediction times from approximately 2,750 and 125 seconds, to approximately 2,250 and 122 seconds, respectively. This indicates the valuable robustness of TSCLAS concerning the missing data problem in time series, a common issue present in real-world IoT data.

6.2.3 Classification of 1-Hour Time Series. To evaluate TSCLAS in a broader sense, we perform the classification of the 1-hour ASOS time series. While the 1-minute ASOS corresponds to time series from airports within the United States, the 1-hour ASOS consists of time series from around

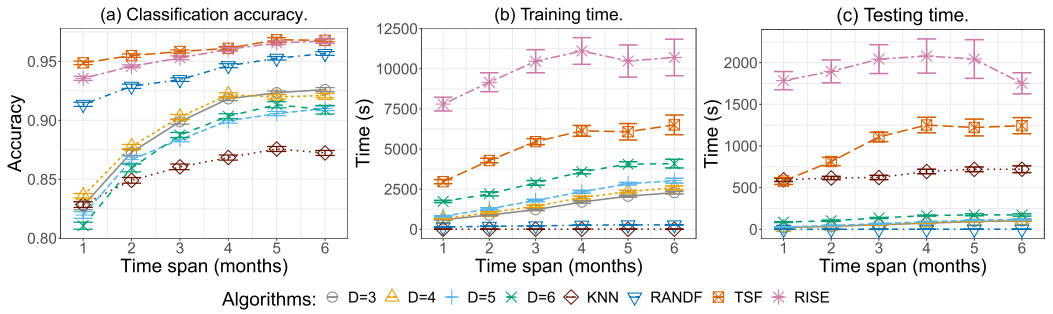


Fig. 9. Results for the experiments on the 1-hour ASOS time series, consisting of (a) the classification accuracy, and times spent on (b) the training and (c) the prediction steps. All results were evaluated considering time spans of 1, 2, 3, 4, 5, and 6 months, with a starting date of January 1, 2020.

the world, as illustrated in Figure 2(b). For this experiment, we consider the time series from all 3,742 airports as a single dataset, performing 30 classifications with randomized splits each, as described in Section 6.1. Figure 9(a) presents the classification accuracies achieved by the classifiers for different time spans, ranging from 1 month to 6 months.

We can see that all classifiers are impacted by the number of observations per time series, but TSCLAS is the most dependent on the time series length. As the length increases, our achieved accuracy also increases. It ranges from the worst accuracy level of 0.81 for $D = 6$ in the 1-month time span, with only 744 observations, to our highest accuracy of 0.93 for $D = 3$ in the 6-month time span, where there are 4,368 observations, as presented in Table 2. However, we can see by the tendency of the curves that we are bounded at this highest accuracy. This behavior occurs due to the low representativeness of the time series dynamics that we can capture from these phenomena by having a fixed 1-hour time interval. Unlike the 1-minute ASOS time series, where lower time intervals, up to 15 minutes, give reasonable representativeness for the phenomena dynamics, the 1-hour intervals are too much spaced. However, as can also be seen, this is not a problem for the others strategies in comparison with ours, where TSF reaches classification accuracies between 0.95 and 0.97, for the 1-month and 6-month configurations, respectively.

By looking at Figures 9(b) and 9(c), which present the time for training and prediction, respectively, we can note that both TSF and RISE need more time to achieve their highest accuracies. On the other hand, KNN, which practically does not have operations to do in the training step, presents a considerable time for the prediction step. TSCLAS presents reasonable times for both steps, being below TSF and RISE in the training step and only losing for the RANDF algorithm in the prediction step. In fact, regarding the trade-off between classification times and achieved accuracy, for all algorithms, RANDF appears as the best choice for this 1-hour time interval scenario, with accuracies increasing from 0.91 to 0.95 and a short processing time at both steps.

In summary, we can conclude that, although TSCLAS is still competitive in this scenario, it is not the most favorable for it. On the contrary, time series with smaller granularities, such as the 1-minute scenarios, are more suitable. In fact, we can also note that, in the current IoT scenarios, it is more likely to have sensors collecting data at small intervals, even in seconds, than at 1-hour intervals. Thus, justifying the main objectives for TSCLAS to be a valuable strategy for the classification of IoT time series data according to their dynamical behavior.

7 CONCLUSION

In this work, we presented TSCLAS, a strategy for time series classification, using features extracted from ordinal patterns transformations, demonstrated to be a very suitable solution for

classifying data generated by IoT sensors. TSCLAS is exceptionally adaptable to time series length and robust to missing data gaps. We conducted several experiments using real-world meteorological time series, leading to significant classification accuracies for both time series from 1-minute and 1-hour intervals between consecutive data points. The method achieved better results when the time series lengths are large enough, and also when the data points provide satisfactory representativeness of the measured phenomena.

Besides that, TSCLAS presents lower computation times for training and testing, being competitive enough compared to the tested algorithms from the literature. By simulating a missing data problem as consecutive gaps in the time series, which is common and likely to occur in IoT scenarios, TSCLAS surpasses the algorithms from literature in our tests, even with gaps as long as 50% of time series data points.

Furthermore, although further studies are needed to confirm it, we also argue that the proposed class separability analysis can be a valuable method to estimate the classification potential of ordinal patterns transformations. Prior data analysis for a given dataset can be helpful as a first step in knowing if its time series classes are distinguishable or not. Another future work includes the application of the proposed classification strategy to other datasets, in other domains, to assure its viability to different scenarios. Finally, we plan to evaluate the robustness of TSCLAS to other IoT common problems, such as precision, correction, and trustworthiness.

REFERENCES

[1] Andre L. L. Aquino, Tamer S. G. Cavalcante, Eliana S. Almeida, Alejandro C. Frery, and Osvaldo A. Rosso. 2015. Characterization of vehicle behavior with information theory. *The European Physical Journal B* 88, 10 (2015), 257.

[2] Andre L. L. Aquino, Heitor S. Ramos, Alejandro C. Frery, Leonardo P. Viana, Tamer S. G. Cavalcante, and Osvaldo A. Rosso. 2017. Characterization of electric load with information theory quantifiers. *Physica A: Statistical Mechanics and Its Applications* 465 (2017), 277–284.

[3] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31, 3 (2017), 606–660.

[4] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. 2015. Time-series classification with COTE: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2522–2535.

[5] Christoph Bandt and Bernd Pompe. 2002. Permutation entropy: A natural complexity measure for time series. *Physical Review Letters* 88, 17 (2002), 174102.

[6] Korkut Bekiroglu, Seshadhri Srinivasan, Ethan Png, Rong Su, and Constantino Lagoa. 2020. Recursive approximation of complex behaviours with IoT-data imperfections. *IEEE/CAA Journal of Automatica Sinica* 7, 3 (2020), 656–667.

[7] João B. Borges, Heitor S. Ramos, Raquel A. F. Mini, Osvaldo A. Rosso, Alejandro C. Frery, and Antonio A. F. Loureiro. 2019. Learning and distinguishing time series dynamics via ordinal patterns transition graphs. *Appl. Math. Comput.* 362 (2019), 124554.

[8] Joao B. Borges, Heitor S. Ramos, Raquel A. F. Mini, Aline C. Viana, and Antonio A. F. Loureiro. 2019. The quest for sense: Physical phenomena Classification in the Internet of Things. In *Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 701–708.

[9] João Borges Neto, Thiago Silva, Renato Assunção, Raquel Mini, and Antonio Loureiro. 2015. Sensing in the collaborative Internet of Things. *Sensors* 15, 3 (2015), 6607–6632.

[10] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.

[11] Thomas Buchholz, Michael Schiffers, Axel Küpper, and Michael Schiffers. 2003. Quality of context: What it is and why we need it. In *Proceedings of the Workshop of the HP OpenView University Association* (Geneve, Switzerland), 1–14.

[12] J. Orestes Cerdeira, M. João Martins, and Pedro C. Silva. 2012. A combinatorial approach to assess the separability of clusters. *Journal of Classification* 29, 1 (2012), 7–22.

[13] Long Chen, Linqing Wang, Zhongyang Han, Jun Zhao, and Wei Wang. 2019. Variational inference based kernel dynamic Bayesian networks for construction of prediction intervals for industrial time series with incomplete input. *IEEE/CAA Journal of Automatica Sinica* 7, 5 (2019), 1–9.

[14] Houtao Deng, George Runger, Eugene Tuv, and Vladimir Martyanov. 2013. A time series forest for classification and feature extraction. *Information Sciences* 239 (2013), 142–153.
[15] Bruna Amin Gonçalves, Laura Carpi, Osvaldo A. Rosso, and Martín G. Ravetti. 2016. Time series characterization via horizontal visibility graph and Information Theory. *Physica A* 464 (September 2016), 93–102.
[16] John Greene. 2001. Feature subset selection using Thornton's separability index and its applicability to a number of sparse proximity-based classifiers. In *Proceedings of the Pattern Recognition Association of South Africa.*
[17] Aimad Karkouch, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. 2016. Data quality in Internet of Things: A state-of-the-art survey. *Journal of Network and Computer Applications* 73 (2016), 57–81.
[18] Christopher W. Kulp, Jeremy M. Chobot, Helena R. Freitas, and Gene D. Sprechini. 2016. Using ordinal partition transition networks to analyze ECG data. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 26, 7 (2016), 73114.
[19] Christopher W. Kulp and Suzanne Smith. 2011. Characterization of noisy symbolic time series. *Physical Review E* 83, 2 (2011), 26201.
[20] C. W. Kulp and L. Zunino. 2014. Discriminating chaotic and stochastic dynamics through the permutation spectrum test. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 24, 3 (2014), 33116.
[21] P. W. Lamberti, M. T. Martin, A. Plastino, and O. A Rosso. 2004. Intensive entropic non-triviality measure. *Physica A: Statistical Mechanics and its Applications* 334, 1–2 (2004), 119–131.
[22] H. A. Larrondo, M. T. Martín, C. M. González, A. Plastino, and O. A. Rosso. 2006. Random number generators and causality. *Physics Letters A* 352, 4–5 (2006), 421–425.
[23] Jason Lines and Anthony Bagnall. 2015. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery* 29, 3 (2015), 565–592.
[24] Jason Lines, Sarah Taylor, and Anthony Bagnall. 2018. Time series classification with HIVE-COTE. *ACM Transactions on Knowledge Discovery from Data* 12, 5 (2018), 1–35.
[25] Jason Lines, Sarah Taylor, Anthony Bagnall, and East Anglia. 2016. HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM)*, Vol. 12. IEEE, 1041–1046.
[26] Caihua Liu, Patrick Nitschke, Susan P. Williams, and Didar Zowghi. 2020. Data quality and the Internet of Things. *Computing* 102, 2 (2020), 573–599.
[27] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Király. 2019. SKTIME: A unified interface for machine learning with time series. *Workshop on Systems for ML at NeurIPS 2019* NeurIPS (2019).
[28] S. Makhija, S. Saha, S. Basak, and M. Das. 2019. Separating stars from quasars: Machine learning investigation using photometric data. *Astronomy and Computing* 29 (2019), 100313.
[29] Kezhi Z. Mao and Wenyin Tang. 2011. Recursive mahalanobis separability measure for gene subset selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8, 1 (2011), 266–272.
[30] Samuel Maurus and Claudia Plant. 2016. Skinny-dip: Clustering in a sea of noise. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, 1055–1064.
[31] Michael McCullough, Michael Small, Thomas Stemler, and Herbert Ho-Ching Iu. 2015. Time lagged ordinal partition networks for capturing dynamics of continuous dynamical systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 25, 5 (2015), 53101.
[32] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. 2018. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 2923–2960.
[33] Federico Montori, Kewen Liao, Prem Prakash Jayaraman, Luciano Bononi, Timos Sellis, and Dimitrios Georgakopoulos. 2018. Classification and annotation of open Internet of Things datastreams. In *Web Information Systems Engineering – WISE 2018 (Lecture Notes in Computer Science)*, Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou (Eds.). Vol. 1. Springer International Publishing, 209–224.
[34] F. Olivares, M. Zanin, L. Zunino, and D. G. Pérez. 2020. Contrasting chaotic with stochastic dynamics via ordinal transition networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30, 6 (2020), 063101.
[35] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[36] Michael Postol, Candace Diaz, Robert Simon, and Drew Wicke. 2019. Time-series data analysis for classification of noisy and incomplete Internet-of-Things datasets. In *Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 1543–1550.
[37] Yongrui Qin, Quan Z. Sheng, Nickolas J. G. Falkner, Schahram Dustdar, Hua Wang, and Athanasios V. Vasilakos. 2016. When things matter: A survey on data-centric Internet of Things. *Journal of Network and Computer Applications* 64 (2016), 137–153.

[38] Chotirat Ann Ratanamahatana and Eamonn Keogh. 2005. Three myths about dynamic time warping data mining. In *Proceedings of the 2005 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 506–510.
[39] Martín Gómez Ravetti, Laura C. Carpi, Bruna Amin Gonçalves, Alejandro C. Frery, and Osvaldo A. Rosso. 2014. Distinguishing noise from chaos: Objective versus subjective criteria using horizontal visibility graph. *PLoS ONE* 9, 9 (2014), e108004.
[40] Haroldo V. Ribeiro, Max Jauregui, Luciano Zunino, and Ervin K. Lenzi. 2017. Characterizing time series via complexity-entropy curves. *Physical Review E* 95, 6 (2017), 062106.
[41] Osvaldo A. Rosso, Laura C. Carpi, Patricia M. Saco, Martín Gómez Ravetti, Angelo Plastino, and Hilda A. Larrondo. 2012. Causality and the entropy-complexity plane: Robustness and missing ordinal patterns. *Physica A: Statistical Mechanics and its Applications* 391, 1–2 (2012), 42–55.
[42] O. A. Rosso, H. A. Larrondo, M. T. Martin, A. Plastino, and M. A. Fuentes. 2007. Distinguishing noise from chaos. *Physical Review Letters* 99, 15 (2007), 154102.
[43] Osvaldo Anibal Rosso, Felipe Olivares, and Angelo Plastino. 2015. Noise versus chaos in a causal Fisher-Shannon plane. *Papers in Physics* 7, (2015), 070006.
[44] Osvaldo A. Rosso, Felipe Olivares, Luciano Zunino, Luciana De Micco, André L. L. Aquino, Angelo Plastino, and Hilda A. Larrondo. 2013. Characterization of chaotic maps using the permutation Bandt-Pompe probability distribution. *The European Physical Journal B* 86, 4 (2013), 116.
[45] Osvaldo A. Rosso, Raydonal Ospina, and Alejandro C. Frery. 2016. Classification and verification of handwritten signatures with time causal information theory quantifiers. *PLOS ONE* 11, 12 (2016), e0166868.
[46] O. A. Rosso, L. Zunino, D. G. Pérez, A. Figliola, H. A. Larrondo, M. Garavaglia, M. T. Martín, and A. Plastino. 2007. Extracting features of Gaussian self-similar stochastic processes via the Bandt-Pompe approach. *Physical Review E* 76, 6 (2007), 061114.
[47] P. Sanchez-Moreno, R. J. Yanez, and J. S. Dehesa. 2009. Discrete densities and fisher information. In *Difference Equations and Applications*. 291–298.
[48] Patrick Schäfer. 2015. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery* 29, 6 (2015), 1505–1530.
[49] Omer Berat Sezer, Erdogan Dogdu, and Ahmet Murat Ozbayoglu. 2018. Context-aware computing, learning, and big data in internet of things: A survey. *IEEE Internet of Things Journal* 5, 1 (2018), 1–27.
[50] Taciano Sorrentino, C. Quintero-Quiroz, A. Aragoneses, M. C. Torrent, and Cristina Masoller. 2015. Effects of periodic forcing on the temporally correlated spikes of a semiconductor laser with feedback. *Optics Express* 23, 5 (2015), 5571.
[51] Jinjun Tang, Yinhai Wang, and Fang Liu. 2013. Characterizing traffic time series based on complex network theory. *Physica A: Statistical Mechanics and its Applications* 392, 18 (2013), 4192–4201.
[52] R. Core Team. 2018. *A Language and Environment for Statistical Computing*. (2018), https://www.R-project.org.
[53] Chris Thornton. 1998. Separability is a learner's best friend. In *Proceedings of the 4th Neural Computation and Psychology Workshop, (London, U.K., April 9–11, 1997)*, John A. Bullinaria, David W. Glasspool, and George Houghton (Eds.). Springer, London, 40–46.
[54] Chun-Wei Tsai, Chin-Feng Lai, Ming-Chao Chiang, and Laurence T. Yang. 2014. Data mining for Internet of Things: A survey. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 77–97.
[55] Giora Unger and Benny Chor. 2010. Linear separability of gene expression data sets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 7, 2 (2010), 375–381.
[56] Lei Wang and Lei Wang. 2008. Feature selection with kernel class separability. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 9 (2008), 1534–1546.
[57] Yulei Wu. 2021. Robust learning-enabled intelligence for the Internet of Things: A survey from the perspectives of noisy data and adversarial examples. *IEEE Internet of Things Journal* 8, 12 (2021), 9568–9579.
[58] Jie Zhang, Junfeng Sun, Xiaodong Luo, Kai Zhang, Tomomichi Nakamura, and Michael Small. 2008. Characterizing pseudoperiodic time series through the complex network approach. *Physica D* 237, 22 (2008), 2856–2865.
[59] Jiayang Zhang, Jie Zhou, Ming Tang, Heng Guo, Michael Small, and Yong Zou. 2017. Constructing ordinal partition transition networks from multivariate time series. *Scientific Reports* 7, 1 (2017), 7795.
[60] Hongwen Zheng and Yanxia Zhang. 2008. Feature selection for high-dimensional data in astronomy. *Advances in Space Research* 41, 12 (2008), 1960–1964.
[61] L. Zunino, Miguel C. Soriano, and O. A. Rosso. 2012. Distinguishing chaotic and stochastic dynamics from time series by using a multiscale symbolic approach. *Physical Review E* 86, 4 (2012), 46210.

Received April 2021; revised December 2021; accepted April 2022