TimeCIEL: Contextual Interactive Ensemble Learning for Time Series Classification

Jordan Levy, Clément Blanco-Volle, Nicolas Verstaevel, Benoit Gaudou, Vincent Talon

TimeCIEL : Contextual Interactive Ensemble Learning for Time Series Classification

Jordan Levy[1,2][0009−0002−4337−2608], Clément Blanco-Volle[1,3][0009−0000−8017−4702], Nicolas Verstaevel[1][0000−0002−7879−6681], Benoit Gaudou[1][0000−0002−9005−3004], and Vincent Talon[2]

[1] Institut de Recherche en Informatique de Toulouse, Toulouse, France
{first.last}@irit.fr
[2] TwinswHeel, Fontanes, France
[3] Continental Digital Services France, Toulouse, France

Abstract. Multivariate time series classification is a challenging task where black box models achieve high performances. However, in real-world applications, interpretability is crucial for helping users understand the decision-making process of an algorithm, not just its performance. In this paper, we present a multi-agent ensemble learning approach for time series classification suited for online learning. Our approach relies on the organization of agents in the feature space at each time steps. We demonstrate that our approach achieves performances comparable to state-of-the-art methods. Finally, we highlight its explainability and interpretability properties as a white-box model.

Keywords: Classification · Ensemble learning · Supervised learning · Time series · XAI

1 Introduction

Classification of time series is an important task which can be useful in various domains such as finance, healthcare or robotics [14]. Over the past decade, many approaches were developed in order to achieve the best performances for this task [12]. However, these advancements often come at the cost of increased model complexity, leading to reduced explainability.

While performance is undoubtedly important, interpretability plays a major role in helping users understand the decision-making process of an algorithm. A model that provides clear explanations enhance user trust and acceptance, making the system more reliable and transparent [14].

Time series classification is a difficult task, as it requires the identification of patterns not only across feature dimensions, but also over the time axis. This complexity is usually handled by black-box models that increase the lack of interpretability, which can be dangerous in the case of prediction errors. In contrast, white-box models offer greater ease of interpretation, enabling users to understand the sources of error and refine the classification process.

In this paper, we present a time series multi-agents learning system for supervised learning [4]. Agents are temporalised and spatialised in the feature space to encompass time series. Our approach is specifically designed for online learning with great capacities of explainability.

The main contributions of this article are a new approach for time series classification, a comparison of our approach with the best and most popular approaches in the literature and explainability metrics that demonstrate the benefits of our model.

The rest of the paper is presented as follows. Section 2 presents the related work to our approach. Section 3 details our multi-agent system. Section 4 is dedicated to the comparison of our model. Finally, Section 5 demonstrates the interpretability capabilities of our system.

2 Related work

In this section, we review existing methods related to our approach to provide a comprehensive context for our contributions.

2.1 Multivariate time series classification

Time series classification refers to the task of classifying a time series $X_t = [x_0, x_1, ..., x_t]$ of t time steps. Specifically, multivariate time series classification (MTSC) refers to the case where at each time step, the time series has n features. So, $x_t \in \mathbb{R}^n$.

One of the most popular approaches for time series classification is the metric of similarity Dynamic Time Wrapping (DTW) combined with K-nearest neighbor to perform classification [12]. In particular, for MTSC, the two main strategies are treating each feature dependently (DTW_D) or independently (DTW_I) [13]. Our approach is also using distances in the feature space.

One of the best approaches in both performance and computation efficiency [12] is ROCKET [5]. Their approach consists of using a large number of random convolutional kernels on the multivariate time series, and after pooling, applying a ridge regression classifier.

Ensemble learning has also been explored for MTSC [7], [10], and has shown great performances [12]. In ensemble learning, an ensemble of learners are making a prediction of the same time series and with an aggregation function, a new result is computed. *InceptionTime* [7] is a deep learning approach which uses an ensemble of Inception networks based on convolutional networks of different lengths. $HIVE-COTE$ 2.0 [10] uses 4 state-of-the-art adapted classifiers. Our approach is also an ensemble learning model but only uses weak learners to remain a white box.

[4] Our code is available at: https://github.com/jordanlv/TimeCIEL

2.2 Learning with hyper-rectangles

Learning with hyper-rectangles is an approach already explored in the literature for tabular data [3], [6], [8]. Usually, these approaches aim to partition the feature space into hyper-rectangles, with each hyper-rectangle corresponding to an internal model able of making predictions.

In [6], the authors shows the capacity of an ensemble model based on hyper-rectangles to perform non-linear classification tasks while using linear internal models (SVM, etc.). Similarly, in [3], the authors demonstrated the performances of a similar model but for regression tasks while showing the capacity of explainability of their model. Finally, in [8], the authors use gradient boosting to learn their hyper-rectangle bounds.

To the best of our knowledge, our approach is the first to use hyper-rectangles for time series classification. Using hyper-rectangles for time series classification can offer advantages, such as improved interpretability and online learning.

2.3 Time series explainability

Explainability is a well-known domain with many contributions in recent years, especially for tabular data [4]. Explainability methods are usually categorized as global or local and as model-specific or model-agnostic [4]. Global methods explain a model in its entirety, while local methods focus on explaining a specific predicted instance. Specific methods are tailored to a particular model, whereas agnostic methods can be applied to any model.

Time series explainability refers to explanations specifically designed for time series data, taking into account its temporal dependencies. The same classification of methods is used in time series explainability [14].

$TimeSHAP$ [2] is a model agnostic for both local and global explanations for time series. This post-hoc approach uses Shapley values, which can be applied to any model to generate local explanations that can be aggregated to form a global explanation.

However, some researchers argue that instead of combining black-box models with explainability methods, we should rely on inherently interpretable models for critical decision-making [11].

3 TimeCIEL

This section introduces TimeCIEL, derived from [3] and so is also using hyper-rectangles. However, while the model in [3] is suited for tabular data, our model can be used for time series by adding a temporal dimension. In section 3.1, we first describe the operations on hyper-rectangles that our system uses. Next, in section 3.2, we present how we encompass time series with hyper-rectangles.

3.1 Operations with hyper-rectangles

Before introducing our system, we define some operations related to hyper-rectangles used in [3]. Hyper-rectangles are n-dimensional rectangles. They are defined by a lower bound and a higher bound on each of their n dimensions:

$$H = [l_1, h_1] \times [l_2, h_2] \times \cdots \times [l_n, h_n]$$

When using hyper-rectangles, we usually want to select them to update them. The following definitions introduce two selection operators which are also illustrated in Figure 1.

Definition 1. *A hyper-rectangle H is activated by a point $x \in \mathbb{R}^n$ if x is inside H:*

$$activated(H, x) = \bigwedge_{i=1}^{n} l_i \leq x_i \leq h_i$$

Definition 2. *A hyper-rectangle H is in the neighborhood of a point x if the hyper-rectangle representing its neighborhood N intersect H:*

$$neighbor(H, N) = \bigwedge_{i=1}^{n} max(H_{l_i}, N_{l_i}) \leq min(H_{h_i}, N_{h_i})$$

The hyper-rectangle representing the neighborhood of a point $x \in \mathbb{R}^n$ is the hyper-rectangle of center x with its size on each dimension defined by the hyperparameter $R \in \mathbb{R}^n$.

$$N = [x_1 - R_1, x_1 + R_1] \times \cdots \times [x_n - R_n, x_n + R_n]$$

One of the main reasons of using hyper-rectangles is that their dimensions can easily change independently of the other dimensions. Indeed, in [3], hyper-rectangles are able to extend or retract by a factor of α. If a hyper-rectangle H extends (resp. retracts) itself in a direction of a point $x \in \mathbb{R}^n$, then its higher bounds and lower bounds are updated:

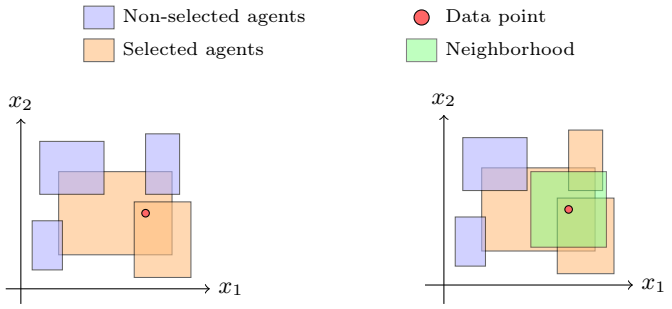
$$h_i = \begin{cases} (h_i - l_i)\varepsilon^{\frac{1}{\theta}} + l_i & \text{if } l_i \leq x_i \leq h_i \\ h_i & \text{else.} \end{cases}$$

$$l_i = \begin{cases} (l_i - h_i)\varepsilon^{\frac{1}{\theta}} + h_i & \text{if } l_i \leq x_i \leq h_i \\ l_i & \text{else.} \end{cases}$$

with

$$\varepsilon = \begin{cases} (1 + \alpha) & \text{if extending} \\ (1 - \alpha) & \text{if retracting} \end{cases}$$

and θ the number of dimensions of x where $l_i \leq x_i \leq h_i$. With this formula, the volume of H is multiplied by a factor of α.



(a) activated (b) neighbor

Fig. 1: Example in 2 dimensions of activated agents and of neighbor agents. In 1a, the point activates two agents because it lies between their bounds on each dimension. In 1b, the point activates three agents because its neighborhood intersects these three agents.

3.2 TimeCIEL

Our system is composed of spatiotemporal agents. A spatiotemporal agent A is composed of an ensemble of hyper-rectangles H_t and an internal model $f(\cdot)$. Even though we can put any internal model in our agent like in [6], we decided that agents predict only one class y. Hyper-rectangles are temporalized at the frequency of the n-dimensional input time series. Formally, $A = (y, \{H_0, H_1, ..., H_t\})$.

The main objective of the agents is to pave the spatiotemporal space in order to represent the training data while generalizing to new data. So, during each epoch of the training, agents are making a prediction to know if they should update their hyper-rectangles.

Before introducing the learning rules, we introduce two new definitions adapted from definitions 1 and 2 for spatiotemporal agents.

Definition 3. *An agent A is activated by a time series X_t if the time series activates all its hyper-rectangles:*

$$activated(A, X_t) = \bigwedge_{i=0}^{t} activated(H_i, x_i)$$

Definition 4. *An agent A is in the neighborhood of a time series X_t if the time series has some hyper-rectangles of A in its neighborhood. Precisely, we define N_{rate}, a hyperparameter representing the rate of the neighbor that a time series should have to be in the neighborhood of A:*

$$neighbor(A, X_t) = \sum_{i=0}^{t} neighbor(H_i, x_i) > N_{rate} * t$$

From these new definitions, we can adapt the learning rules from [3] to our system:

- An agent A is created when the time series X_t has no neighbor and does not activate any agents. The initial size of the agent is set with a parameter $R \in \mathbb{R}^n$ which defines the initial size of the generalization for each dimension.
- If the time series has at least one neighbor and doesn't activate any agent, we use the mean squared error to determine if the hyper-rectangles of each agent should be extended or retracted. Specifically, hyper-rectangles of an agent extend (resp. retract) only if the points at the same time step are not activated by (resp. in the neighborhood of) the corresponding hyper-rectangles. The mean squared error is used because our approach is able to do batch learning and so we have to compute the mean error for each agent. In the specific case when all agents are doing bad predictions, we retract them but we also create a new agent around this time series. This case means that the system has never seen this type of time series at this position and so we need to create a new agent.
- If the time series activates more than one agent, the system classifies the predictions of the time series and similarly to the case before, if their prediction is bad, the agent retracts, if needed, its hyper-rectangles.
- Finally, the system is able to destroy agents depending on if they are usually doing bad predictions. Indeed, each agent has two score *goods* and *bads* representing the number of times an agent has done good predictions or bad predictions. Then, we introduce a new hyperparameter $T_{destroy}$, representing the threshold when the system should destroy an agent. So, at the end of each epoch, the system checks for each agent if $goods - bads > T_{destroy}$. In the case that this condition is respected, the agent is being destroyed.

After the training, the model can be used to predict multivariate time series through a cascading process : if agents are activated, the most represented class is chosen; If some agents are neighbors to the time series, the majority class is selected; Finally, the class of the nearest agent is chosen, determined by measuring the distance between each point of the time series and hyper-rectangles borders.

A great property of our algorithm is that we can easily modify the architecture even after the first training ended. Indeed, we can still create, delete and update an agent as we want. This means that we can always start a new training and so, our model is able to do online learning. This property is useful when an expert wants to give feedback during the exploitation period of the model.

4 Benchmark

To evaluate our system, we compare it to other approaches using a benchmark of 10 datasets [12]. The original benchmark included 26 datasets, but we selected the first 10 with less than 100 dimensions and less than 2000 time steps, as our

model is not suited for high-dimensional data because of the curse of dimensionality. If the length of the time series is too long, we recommend splitting the time series in multiple splits. Table 1 shows the characteristics of each datasets used. We invite the reader to read the benchmark for more description of each dataset.

Table 1: Dataset characteristics

	Code Name	Train Size	Test Size	Dims	Length	Classes
AWR	ArticularyWordRecognition	275	300	9	144	25
AF	AtrialFibrillation	15	15	2	640	3
BM	BasicMotions	40	40	6	100	4
CR	Cricket	108	72	6	1197	12
EP	Epilepsy	137	138	3	206	4
EC	EthanolConcentration	261	263	3	1751	4
ER	ERing	30	270	4	65	6
FM	FingerMovements	316	100	28	50	2
HMD	HandMovementDirection	160	74	10	400	4
HW	Handwriting	150	850	3	152	26

Following the original benchmark, we perform 30 stratified resamples, with the first resample using the original train/test split and subsequent ones seeded by the resample number. As recommended, we do not scale the data. In the original benchmark, the authors performed external tuning if in the original paper, the algorithm was tuned. We decided to perform external tuning in our approach since the initial size of hyper-rectangles is a very sensitive parameter. To do so, we use a nested cross-validation approach: splitting the training set into three, training on two, and validating on the third to find optimal hyperparameters. We tune parameters (R, α, T_{bad}, N_{rate}) using Optuna [1] with TPESampler, exploring values between 0.01 and 0.9. The final model is trained with the best average parameters and evaluated on the test split.

Table 2 shows the performances of our algorithm compared to the baseline DTW_D and the top classifiers from the original benchmark: $ROCKET$ and $InceptionTime$. We used aeon [9] to train these models. When doing our benchmark, we couldn't reproduce the exact same results of [12] : the mean difference with their results is 0.465%. Our algorithm has slightly lower performances compared to the best algorithms : the mean difference between DTW_D, $ROCKET$ and $InceptionTime$ and our model is respectively 7.87%, 12.46% and 9.39%.

5 Explainability

Similar to the original model, our system has great capacities of explainability [3]. Indeed, our model is able to give global and local explanation [14]. We can create metrics on the overall architecture of the algorithm, and we can understand why a time series is classified as such.

Table 2: Results

	TimeCIEL (%)	DTW_D (%)	ROCKET (%)	InceptionTime (%)
AWR	97.27±0.82	98.81±0.57	**99.52±0.31**	99.07±0.43
AF	**36.00±7.81**	20.89±5.90	25.33±9.17	21.11±9.12
BM	65.50±4.20	94.58±3.60	98.75±1.80	**100.0±0.00**
CR	94.12±2.97	**100.0±0.00**	**100.0±0.00**	99.63±0.71
EP	71.86±5.32	96.50±1.39	**99.01±0.94**	98.60±0.99
EC	30.08±2.57	31.13±2.14	46.55±2.15	28.94±2.28
ER	91.54±3.33	93.35±1.24	**98.00±0.69**	91.80±2.10
FM	53.93±4.85	54.77±4.72	58.20±4.71	**58.70±3.48**
HMD	29.28±3.46	30.14±5.86	**44.82±5.14**	38.02±5.42
HW	32.82±2.14	**60.95±2.29**	56.80±2.17	60.45±2.42

5.1 Global

Our model is a white-box model as agents are just a series of hyper-rectangles. From this, it is possible to understand how the model works. Specifically, we define the hyper-rectangle density metric $D(x, H_c)$ for a class c as the number of hyper-rectangles covering a value x. Formally,

$$D(x, H_c) = \sum_{h \in H_c} activated(h, x)$$

where H_c are all the hyper-rectangles in the model of class c. This metric can help us to determine the main characteristics of a class. Figure 2 shows, for the dataset UWaveGestureLibrary[5], the characteristics of a class on a single dimension and for all time steps on a heatmap where yellow areas highlight the most representative parts of the class.

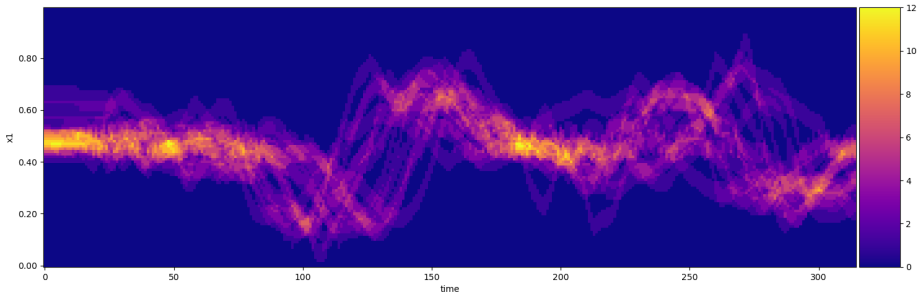


Fig. 2: Heatmap of the hyper-rectangle density on the first dimension of the second class. The most representative parts of this class are between time steps 0-50 and 170-200.

[5] Dataset available at: https://www.timeseriesclassification.com

5.2 Local

A local explanation is an explanation on a single time series. Specifically, in time-series classification, we can divide the explanation in three levels: time points based, subsequences based and instance based [14]. The three levels depend on which part of the time series the explanation is given. Our model is able to give an explanation for the three levels.

In this section, we use a simple dataset of three sine waves with different periods, consisting of 100 samples, each with 15 time steps and 1 dimension. The objective of the classification task is to classify each period. Figure 3 shows the dataset. With our model, we achieve the accuracy of 100% with 3 agents: an agent per class.

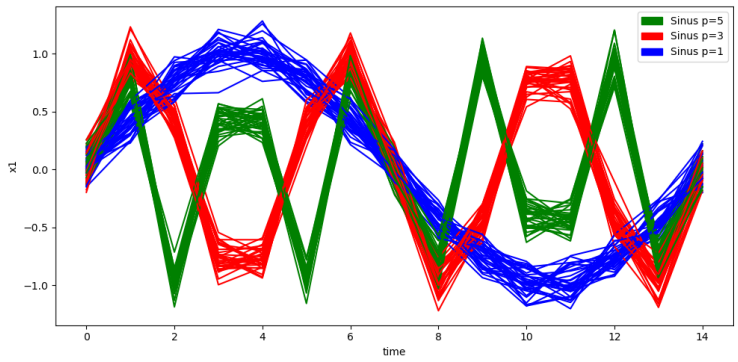


Fig. 3: Dataset of sine waves with periods 1, 3, and 5.

Time points based First, our model is able to give time points based explanation. To perform this explanation, let a time series X_t that we want to explain, we retrieve each class of each agent if their hyper-rectangles at time t is a neighbor to x_t. Simply, we check at each time step if the current point of the time series is close to a hyper-rectangle. From this, we can know for each point of the time series what does the model can predict about the point by taking the class of agents containing neighboring hyper-rectangles. Figure 4 shows the time points based explanation of a sine wave of period 1 of our model trained on the simple dataset. We can see at each time step which class the model could have predicted only based on each individual point.

This explanation can help the user to understand the characteristic points of the time series. However, this explanation does not take into account the temporality of the time series. So, we can adapt this metric to subsequence based explanation.

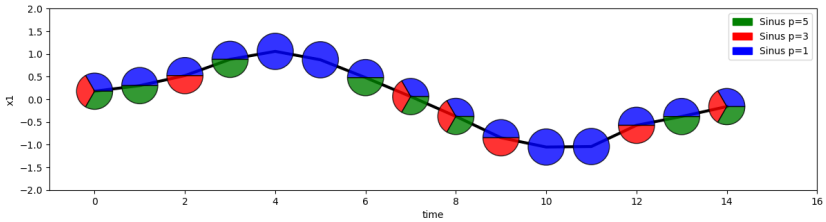


Fig. 4: Time points based explanation of a sine wave of period 1.

Subsequence based The goal of subsequence based explanation is to give an explanation on a subsequence and so, take into account the time dependency between points. Following the time point based explanation, we can compare a subsequence of a time series to the subsequence of each agent at the same time steps. From this, we can check which subsequence of hyper-rectangles is a neighbor to the subsequence we would like to explain. We then take the class of the corresponding agents and we can know the class that are characteristic of this subsequence. Figure 5 shows two explanations of two subsequences. This type of explanation can be useful to identify subparts of a time series responsible for the classification outcomes.

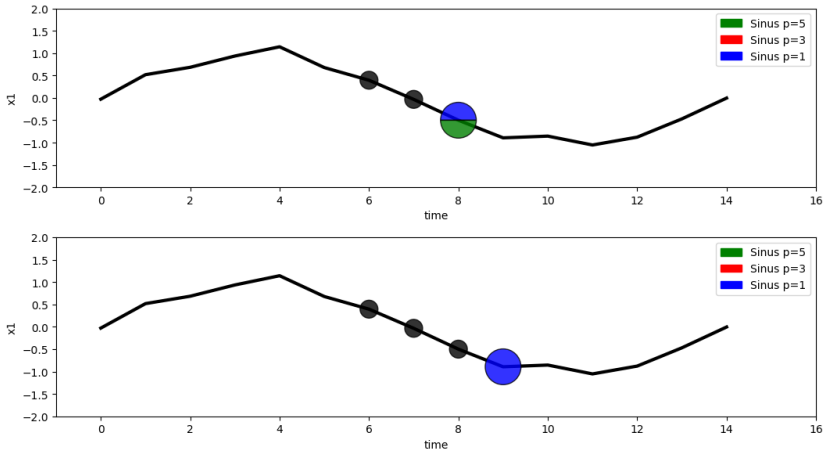


Fig. 5: Subsequence based explanation of a subsequence of a sine wave of period 1. Black points represent the points except the last in the subsequence. The top figure illustrates that with 3 points: the model can only determine that the sine wave is of period 1 or 5. However, in the bottom figure, with a fourth point, the model knows that this subsequence is a sine wave of period 1.

Instance based explanation Finally, our model is able to give instance based explanation which means on the whole time series. By applying the subsequence based explanation on each subsequence from the first point to the last point of the time series, we can determine when the model has done its predictions. Figure 6 illustrates this explanation. With this explanation, we can identify at which point the model has determined the class of the time series.

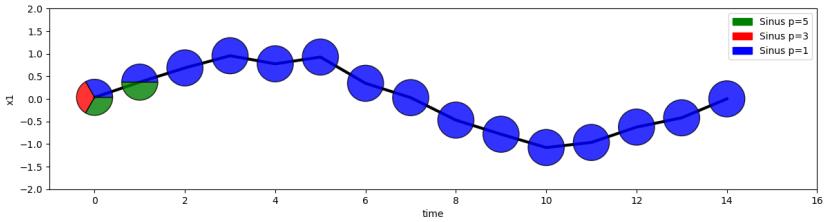


Fig. 6: Instance based explanation. In the simple dataset, the model can determine the type of this time series from the 3rd point.

6 Conclusion

We have presented a time series multi-agent learning system for supervised learning. We have demonstrated that this system achieves comparable performances to state-of-the-art methods, showcasing its ability to model time series data effectively. Additionally, we have shown that our approach enables the system to explain its decisions both spatially and temporally. These conclusions highlight the potential of this multi-agent model for time series classification and explainability. As models become increasingly opaque, our approach offers a white-box solution well-suited for online learning.

However, further work is needed to improve the algorithm's performance and accuracy. It would be interesting to incorporate DTW as a metric by modifying the neighbor and activation functions to account for the optimal alignment determined. Also, even if we have explored some explainability metrics, we believe that our white-box model has even greater potential for interpreting its decisions. Indeed, further work is needed in order to have explainability also based on features for each time step. For now, we still don't know which features have the most influence.

Acknowledgments. We thank the National Association for Research and Technology, the Institut de Recherche en Informatique de Toulouse and the company TwinswHeel for funding of the thesis. We also thank all the reviewers for their help and advice.

Disclosure of Interests. Jordan Levy has received research grants from the company TwinswHeel, owned by Vincent Talon. Clément Blanco-Volle has received research grants from Continental Digital Services France.

References

1. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.
2. João Bento, Pedro Saleiro, André F. Cruz, Mário A. T. Figueiredo, and Pedro Bizarro. TimeSHAP: Explaining Recurrent Models through Sequence Perturbations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2565–2573, 2021.
3. Clément Blanco-Volle, Nicolas Verstaevel, Stéphanie Combettes, Marie-Pierre Gleizes, and Michel Povlovitsch Seixas. Explainability and interpretability of an ensemble multi-agent system for supervised learning. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 335–350. Springer, 2024.
4. Nadia Burkart and Marco F. Huber. A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.
5. Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. 34(5):1454–1495.
6. Thibault Fourez, Nicolas Verstaevel, Frédéric Migeon, Frédéric Schettini, and Frédéric Amblard. How to Solve a Classification Problem Using a Cooperative Tiling Multi-agent System? In Frank Dignum, Philippe Mathieu, Juan Manuel Corchado, and Fernando De La Prieta, editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection*, pages 166–178. Springer International Publishing.
7. Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. InceptionTime: Finding AlexNet for time series classification. 34(6):1936–1962.
8. Andrei V. Konstantinov and Lev V. Utkin. Interpretable ensembles of hyper-rectangles as base models. 35(29):21771–21795.
9. Matthew Middlehurst, Ali Ismail-Fawaz, Antoine Guillaume, Christopher Holder, David Guijo-Rubio, Guzal Bulatova, Leonidas Tsaprounis, Lukasz Mentel, Martin Walter, Patrick Schäfer, and Anthony Bagnall. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024.
10. Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. HIVE-COTE 2.0: A new meta ensemble for time series classification. 110(11–12):3211–3243.
11. Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. 1(5):206–215.
12. Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. 35(2):401–449.
13. Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang, and Eamonn Keogh. Generalizing DTW to the multi-dimensional case requires an adaptive approach. 31(1):1–31.
14. Andreas Theissler, Francesco Spinnato, Udo Schlegel, and Riccardo Guidotti. Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions. 10:100700–100724.