Time Series Classification Using Images: the Case of SAX-like Transformation

Miłosz Wrzesień
the Faculty of Electrical Eng., Automatics, Computer Science and Biomedical Eng.
the AGH University of Science and Technology
Kraków, Poland *wrzesiel@student.agh.edu.pl*

Mariusz Wrzesień
the Faculty of Applied Information Technology
the University of Information Technology and Management
Rzeszów, Poland *mwrzesien@wsiz.edu.pl*

Władysław Homenda
the Faculty of Mathematics and Information Science
the Warsaw University of Technology, Warsaw, Poland
and
the Faculty of Applied Information Technology
the University of Information Technology and Management
Rzeszów, Poland *wladyslaw.homenda@pw.edu.pl*

Abstract

This study concerns the classification of univariate time series. The essence of the survey is transforming time series into two-dimensional monochromatic images. Then, obtained images are classified using convolutional neural networks. Transformation of time series to images is performed in two steps. First, a time series is turned into a string of symbols from an assumed alphabet utilizing SAX-like transformation. The length of the string is supposed to be the square of a natural number. Second, the string of symbols is turned into a square matrix of size equal to the square root of the length of the string representing the time series. Then, each symbol of the matrix is turned into a square-shaped piece of pixels of a grey level determined by the symbol. So then, this operation results in an image (still of square shape) composed of squares of grey pixels. Finally, convolutional neural networks are employed to classify such images. An overall design process is presented with a focus on investigating time series-to-image two-step transformations. Experimental studies involving publicly available data sets are reported, along with an adequate comparative analyses.

Keywords: time series classification, SAX transformation, time series-image transformation, convolutional neural networks

1. Introduction

As the recent interest about time series analysis grows, the excessive number of novel approaches is being studied and developed. Researchers focus on providing more computationally efficient and precise methods of predicting and classifying this type of data. The importance of this domain can be proved with many applications of these models in real-life scenarios like stocks predictions, forecasting the progress of heart diseaseor detecting and classifying fraudulent behaviour in credit cards usage.

The rising interest in this domain motivated us to introduce a novel three-step classification method based on visual representation of the time series. Samples are discretized with *Piecewise Aggregate Approximation* and then transformed into words of defined length and alphabet

with the use of *Symbolic Aggregate Approximation* method. The obtained series of letters are transformed into an image by using certain arrangement and monochromatic color pallet. The classification of generated visuals is performed by specifically designed architecture of Convolutional Neural Network (CNN). The possibility of modifying hyperparameters at different stages of transformations appears to have advantage while conducting fine tuning of the model.

The proposed method can be understood as the one-way image-based encoding process as the numerical time series values are directly transformed into their visual representation. The method allows to generate the image of univariate sample of any size. It can be said that it further eliminates the issue of increasing computational time of longer time series, as it is possible to store every sample in the image size of for example $360x360$ pixels. The total execution time mostly depends on the amount of training and testing samples of selected dataset.

The contribution of this paper into time series classification field can be summarized as:

- Introduction of an efficient transformation of univariate time series samples of any characteristic into their monochromatic visual representation.
- Thorough study of possible model configurations, including the image size, different symbolic word lengths, variety of alphabets and their direct impact on classification outcome.
- General remarks about most favorable settings of the proposed model

2. Literature review

Time series classification is a widely known supervised learning problem. There are multiple domains that take advantage of performing this type of analysis. The examples contain applications like: detecting heartbeat arrhythmia [1], classifying sensor data readings from IoT devices [2], detecting different weather conditions including earthquakes [3], and many more. Time series classification methods can be split into many different groups based on the core assumptions of each method. Two popular categories that are widely studied by researchers are distance based and dictionary based approaches. The method proposed in this article can be understood as a mixture of mentioned approaches. It changes the representation of initial raw time series into words, which are further transformed into images. The success of classification depends on evaluating similarity of the obtained visual representations.

There are many approaches that rely on the distance between two time series. This specifically defined metric is the direct reflection of the similarity of two series. One of the most popular and well performing distance based approaches is *discrete time warping* (DTW) [4]. Shapelet transform [5] and its multiple modifications should also be mentioned.

The core concept of the dictionary based approaches is the change of the representation of the input data and further classification with the use of the obtained features. Usually the discretization method of choice is being applied to the time series intervals of specified size. The so called moving window is then converted into one of the unique symbols from defined alphabet. Many different univariate time series classification approaches including Bag of SFA symbols (BOSS) [6], Contractable BOSS [7], WEASEL [8] has achieved dominant results. MUSE algorithm extends WEASEL method and performs classification on multivariate datasets.

Despite the fact that our approach takes advantage of the dictionary based change of representation the most relevant methods are the ones relying on visual representation of data. The most discussed part of these methods is the way of converting time series into images. Researchers use different transformations to achieve this result. For example, Gramian Angular Fields has been found useful by Wang and Oates [9]. The time series imaging based on so-called lags which represent the difference in succeeding values are thoroughly studied by Homenda et al. [10]. Statistical method of visualizing the recurring nature of states (reccurence plot) has been used as well by Hatami et al. [11] for this purpose. These approaches require the robust classification step which is often performed with Convolutional Neural Networks [12].

3. The method

Presented method consists of three main steps: (i) obtain SAX word from time series, (ii) get visual representation, (iii) perform classification with CNN. This section contains the detailed description of each one of them.

3.1. SAX as the preprocessing method

It has been proven multiple times that the use of suitable time series preprocessing can be both beneficial to computational efficiency and overall models performance. This advantage of modifying time series representation led us to choosing one specific transformation. There were many different techniques to choose from but one particular seemed beneficial to our case. Symbolic aggregate approximation turns the time series of arbitrary length into a string of arbitrary length. Discretization method proposed by the authors uses intermediate representation between the time series itself and the final word. The first step of this process is obtaining Piecewise Aggregate Approximation representation of the raw time series. After initial transformation, obtained values are then symbolized into the output string. Main advantages of this approach is of course dimensionality reduction and lower bounding.

Piecewise Aggregate Approximation. In general, Piecewise Aggregate Approximation, discretizes raw time series with the average values of a moving window with specified length. With this technique, given time series of length n: $C = \{c_1, c_2, ..., c_n\}$ can be reduced to a vector of length w (where $w \leq n$): $\overline{C} = \{\overline{c_1}, \overline{c_2}, ..., \overline{c_w}\}$. The value of the ith element is calculated with an equation:

$$\overline{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j$$

The standardization of the initial time series is worth noting. We want to make sure that the input vector has the mean equal to zero and standard deviation equal to one. With this assumption we are sure that the time series with different peak values and amplitudes are not directly compared.

Symbolic representation. After performing discretization of the given time series, the symbolic representation process is applied. The goal is to transform discrete values into alphabetical symbols. As described by the authors of [13], the advantage of used discretization method is that the numerical values are transformed into symbols with equiprobability. The final symbolic values (letters) are determined based on the so called breakpoints, which are statistically calculated for previously specified alphabet length (number of unique symbols).

Distance measures. The symbolic aggregate approximation has specifically defined distance measures, which allows to further calculate the similarity of two different representations. However, in our approach it is not required to use the predefined metrics, as we propose another way of comparing two symbolic representations.

3.2. Obtaining visual representation

The core advantage of our method is the further transformation of already modified representation of the raw time series. The words obtained in the previous step are converted into images in a specific manner. Each word consists of predefined number of letters from selected alphabet. The final visual representation is made out of squares which correspond to each letter belonging to the generated SAX word. Every letter has its own representation in the form of square in the color of specified gray scale intensity. The general technique of assigning the color to a single letter is directly based on the size of the alphabet used. Gray scale pixels corresponding to letters must take integer values from $[0, 255]$ interval. Value 0 (black color) and the maximum value of 255 (white color) are always assigned to first and last letter of the alphabet, respectively. Corresponding values for letters in between are equally spaced and based on the total number

of letters in the alphabet. For example if we are using 4-letter alphabet, then the corresponding values could be presented as the following dictionary: $a : 0, b : 85, c : 170, d : 255$. If the division without reminder is not possible, then the *floor* function is being applied to obtained float values. With such defined method, it is possible to obtain many different representations of one time series, just by defining various number of word sizes and alphabet lengths combinations. Our method assumes that the length of each word must be the power of two of an integer number. It ensures that the output image will take the form of a square.

3.3. Classification with CNN

We have decided to use Convolutional Neaural Networks (CNN) as a classifier in our approach. CNNs were found useful and exceptionally accurate in models from many different domains that include the step of classifying images. Researchers found them suitable for medical study, natural language processing problems, but also in time series prediction and classification.

Important part of our research was deciding which type or architecture of CNN will give us the best possible results. As our training images are not commonly used while training most popular networks, we did not take the advantage of any transfer learning methods. The state-of-the-art convolutional neural networks are also simply too powerful for our relatively small and not too detailed images. During our research we have implemented and evaluated many diverse architectures of CNNs experimenting with different building blocks of the network.

We have used the total of two convolutional layers with the filters of size 8 and 3x3 kernels. Dropout layers applied directly after pooling layers had coefficients equal to 20%. The last part of the network consisted of 64 node dense connection, dropout layer with 0.35 drop rate and the final layer with the softmax activation function. The model was compiled with the sparse categorical crossentropy loss function and *adam* optimizer.

3.4. Parameters of the model

The presented model can be directed with the total of 3 different hyperparameters.

Word length, as mentioned in the previous sections, is constrained to the powers of two of an integer number $(16, 25, 36, ...)$. It directly specifies the number of discrete segments that the initial raw time series is being split to (or eventually number of letters that the output word consists). It can be also understood as the amount of squares that form the output image.

Alphabet size defines the number of unique letters that the SAX representation is being formed from. After the conversion into an image it sets the number of different grey scale shades. For example, if the alphabet size is set to 1, then we obtain the image of one color, which would not let us distinguish the difference between any of transformed time series. The bigger this number is, the more detailed representation is obtained.

Square size indicates the size (in pixels) of each tile that the output visual representation is formed from grid presented in Figure 1. It is clear that the value of this parameter directly impacts the size of generated output image.

The impact that each parameter has on the output image is visualized in Figure 1.

4. Experimental analysis

18 univariate time series collections from `https://www.timeseriesclassification.com` website were analysed. They represent both binary and multiclass problems with different sample lengths and characteristics. Basic parameters of all datasets that were used in model evaluation are outlined on this website. In the following part of the paper we used abbreviations for datasets: DistalPhalanxOutlineAgeGroup (DistPOAG), DodgerLoopGame (DodgerLoopG), DodgerLoopWeekend (DodgerLoopW),
GunPointAgeSpan (GunPointAS) and GunPointMaleVersusFemale (GunPointMVF).

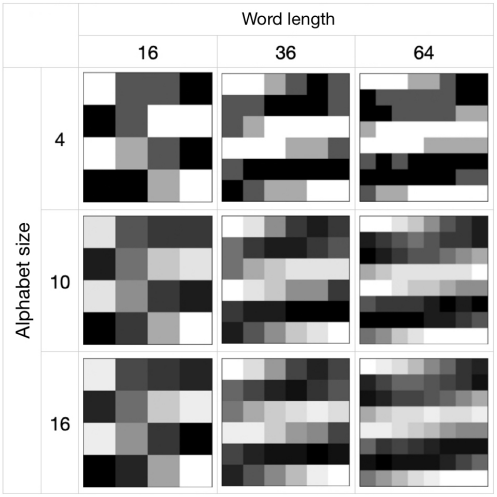

Fig. 1. Parameters of the model and their direct impact on the visual representation of time series.

Experiments have been performed with many different settings of the proposed classifier. After selecting the appropriate network architecture, we have checked numerous hyperparameters combinations, c.f. 3.4. All required calculations were implemented and performed in Python programming language. Results of performed experiments and comparison with the state-of-the-art algorithms are presented in sections 4.1 and 4.2.

4.1. Performed experiments

According to proposed approach, described in Section 3, we have performed similar model evaluations with the use of selected datasets (listed further in tables). The initial stage of the experiments was mainly focused on defining the best possible intervals of the available hyperparameters. This phase concluded with selecting the following values:

- *word length*: $\{16, 25, 36, 49, 64\}$
- *alphabet size*: $\{8, 12, 18, 24\}$
- *square size*: $\{5, 10, 15, 20\}$

The impact of different *word length* values on obtained results is presented in Table 1. All settings have carefully evaluated with 10-fold cross validation. Results suggest that word length has different influence on the classification accuracy depending on the evaluation dataset. In two cases in specific *Chinatown* and *SyntheticControl* which samples' lengths were lower than considered parameter (24 and 60 respectively) , the performance drastically decreased as word length was increasing. As parameter value rises to 64 from 49, the average accuracy on *SyntheticControl* dataset dropped from $94,64\%$ to $16,67\%$. We have also observed samples in which word length was not that impactful. *DodgerLoopWeekend* for instance remained with similar results across all tested parameter values. The last row of 1 tables contains the average accuracy on 18 analyzed datasets for specific parameter configuration. Numbers in bold present the best results across the selected model setting. The difference in evaluation results of parameter settings (in this case 25 and 64) reaches almost 6 percentage points.

Next model parameter that we took into thorough consideration was alphabet size. Further experiments were performed for previously mentioned integer interval of $8, 12, 18, 24$. Table 1 presents detailed comparison of obtained results. Models with alphabet size of 18 and 24 were usually outperforming models with other settings, when considering all evaluation datasets. The difference in average accuracy of best and worst performing models was less than 3 percentage points (half the difference comparing to previously analyzed parameter).

Square size was the third parameter that we wanted to analyze. After performing multiple different experiments on all selected datasets, we came up with the surprising conclusion.

Table 1. Results (in percents) for different **word length**, **alphabet size** and **square size** values

	word length					alphabet size				square size			
DataSet	16	25	36	49	64	8	12	18	24	5	10	15	20
BeetleFly	80,0	**95,0**	**95,0**	90,0	**95,0**	90,0	**95,0**	90,0	**95,0**	90,0	**95,0**	**95,0**	**95,0**
BME	**98,7**	98,0	78,7	74,0	96,7	96,0	**98,7**	96,7	96,7	94,0	96,0	98,0	**98,7**
Chinatown	**98,5**	72,6	72,6	72,6	72,6	96,8	**98,5**	97,38	98,0	97,1	98,0	**98,5**	98,0
Coffee	89,3	96,4	**100**	**100**	96,4	96,4	**100**	**100**	96,4	96,4	**100**	**100**	**100**
DistPOAG	74,8	**78,4**	75,5	74,8	77,7	74,1	77,7	74,1	**78,4**	74,8	75,5	77,7	**78,4**
DodgerLoopG	84,8	84,1	**87,0**	61,6	61,7	84,1	83,3	**87,0**	84,1	83,3	84,8	86,2	**87,0**
DodgerLoopW	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**	**98,6**
GunPointAS	94,9	95,6	95,6	**95,9**	94,9	95,3	95,6	**95,9**	94,9	95,6	94,9	94,9	94,9
GunPointMVF	97,5	**99,4**	98,7	98,7	98,1	97,8	98,7	98,7	**99,4**	98,1	**99,4**	98,1	99,1
Ham	78,1	**82,9**	77,1	80,0	74,3	79,1	80,0	**81,0**	76,2	76,2	80,0	81,0	**82,9**
Herring	68,8	**71,9**	70,3	67,2	68,8	65,6	67,2	**71,9**	68,8	68,8	67,2	**71,9**	70,3
HouseTwenty	83,2	**86,6**	83,2	79,8	80,7	84,0	83,2	84,9	**86,6**	84,9	84,9	84,9	**86,6**
Lightning2	78,7	78,7	80,3	78,7	**82,0**	78,7	78,7	**82,0**	80,3	78,7	78,7	**82,0**	80,3
Meat	70,0	86,7	**100**	90,0	90,0	**100**	90,0	90,0	**100**	**100**	**100**	**100**	**100**
PowerCons	97,2	97,8	**98,3**	83,3	96,7	97,2	96,7	97,2	**98,3**	97,2	**98,3**	97,8	**98,3**
Rock	88,0	90,0	94,0	**96,0**	**96,0**	94,0	92,0	**96,0**	**96,0**	90,0	**96,0**	94,0	**96,0**
SyntheticControl	**94,7**	93,7	88,7	**94,7**	16,7	93,7	94,0	94,3	93,7	**94,7**	**94,7**	**94,3**	93,3
UMD	**95,8**	84,0	89,6	90,3	88,2	92,4	93,8	**95,8**	**95,8**	82,6	90,3	95,1	**95,8**
Columns/average	87,3	88,3	87,9	84,8	82,5	89,7	90,1	90,6	**91,0**	88,9	90,7	91,6	91,8

Only in very few scenarios this parameter meaningfully affected obtained results in positive way. Based on our experiments and available research [14] we decided to check the following values (representing dimensions of square in pixels): $5, 10, 15, 20$. Results are presented in Table 1 in similar form to the previous experiments. Despite the fact the accuracy was highly correlated with the square size, we have decided not to go above the value of 20. Further increases did not provide significantly better results and the time of calculations was substantially longer. Interesting behavior of the model was observed for these four datasets: *BeetleFly*, *Coffee*, *DodgerLoopWeekend*, *Meat*. The best results were obtained by models with three different parameter values: $10, 15, 20$. The difference in these 3 parameter values across all datasets was equal to only 1%.

Summarizing, when it comes to selecting the proper word length it is important to keep this value greater than the sample length as it has significant impact of models accuracy. We have also observed that having high correlation between word length parameter and sample length does not necessarily mean that the overall performance will increase.

In the case of the selection of the size of the alphabet, it is visible that the smallest values 8 and 10 obtained the least amount of best results across selected datasets. The highest values like $16, 18, 24$ on the other hand, had the best achieved performance on 16 out of 18 data collections.

While selecting the square size it is important to choose rather higher values, as they usually provide noticeably better results. We also have came across the situation where none of the parameters modifications had significant impact on the model performance.

Our main goal was the attempt of finding the best parameters combinations that have the positive impact on the performance on multiple datasets. Due to editing restrictions, we have only presented the best achieved results (Tables 1) instead of average accuracy of the 10-fold cross validation, but the conclusions derived from both evaluations were similar.

4.2. Results

Next section contains thorough comparison of results of our classifier and the state-of-the-art results of other researchers obtained from `https://www.timeseriesclassification.com` website. The comparison is prepared for only 16 of which were available on the mentioned website.

Table 2 contains the direct comparison of obtained results vs. the following methods: TS-CHIEF, HIVE-COTE v1.0, ROCKET, InceptionTime, STC, ResNet, ProximityForest, WEASEL, S-BOSS, cBOSS, BOSS RISE TSF and Catch22. Values outline differences in accuracy be-

Table 2. Best accuracy results of SAX-CNN compared with the state-of-the-art algorithms

	SAX	T-C	H-C	RO	I-T	STC	R-N	P-F	WE	S-B	cB	BO	RI	TSF	Cat
BeetleFly	95,0	-0,8	-1,3	**6,5**	**5,7**	**1,7**	**9,7**	**9,0**	**6,3**	**1,3**	-2,5	**0,7**	**7,8**	**11,7**	**11,0**
BME	98,7	-0,9	**0,4**	-1,1	-1,0	**5,7**	-1,2	-1,2	**3,9**	**12,2**	**20,2**	**12,1**	**20,1**	**2,4**	**8,2**
Chinatown	98,5	**2,4**	**2,3**	**1,9**	**2,1**	**2,2**	**1,5**	**3,7**	**2,8**	**10,6**	**3,8**	**10,8**	**9,7**	**3,3**	**5,1**
Coffee	100	**0,9**	**0,7**	**0,0**	**0,1**	**1,1**	**0,4**	**0,8**	**1,1**	**1,9**	**0,9**	**1,4**	**1,5**	**1,3**	**2,0**
DistPOAG	78,4	-4,4	-4,0	-2,7	**1,9**	-1,2	**0,8**	-1,8	-0,9	-3,7	-2,1	-3,6	-3,7	-2,5	**0,1**
GunPointAS	95,9	-4,1	-3,8	-3,5	-2,5	-0,7	-3,6	-3,8	-2,2	-3,6	-4,0	-3,6	-2,7	-1,9	**1,5**
GunPointMVF	99,4	-0,6	-0,6	-0,6	-0,5	**0,7**	**0,4**	-0,6	0,0	-0,6	-0,5	-0,6	**0,2**	-0,2	**0,0**
Ham	82,9	**2,4**	-1,1	-2,7	-2,2	**1,8**	**2,1**	**4,5**	**0,7**	-0,6	**1,7**	-0,9	**0,9**	**2,9**	**13,5**
Herring	71,9	**12,1**	**10,7**	**9,4**	**9,4**	**8,6**	**12,2**	**14,4**	**11,7**	**11,0**	**14,4**	**12,3**	**12,0**	**11,5**	**16,3**
HouseTwenty	86,6	-10,5	-11,3	-9,7	-8,8	-10,9	-9,0	-7,1	**5,5**	-6,4	-7,4	-9,0	-6,4	**2,8**	-8,1
Lightning2	82,0	**5,1**	**4,6**	**4,3**	**0,3**	**16,1**	**1,9**	-2,9	**19,2**	**1,1**	**2,2**	**0,1**	**13,8**	**5,5**	**7,5**
Meat	100	**1,6**	**1,4**	**1,1**	**1,6**	**3,2**	**0,6**	**1,3**	**2,3**	**1,6**	**2,3**	**1,9**	**1,3**	**1,6**	**5,7**
PowerCons	98,3	**0,4**	-0,9	**2,7**	-0,3	**4,3**	**9,7**	-0,4	**6,4**	**8,6**	**8,6**	**9,3**	**2,5**	-1,0	**9,7**
Rock	96,0	**12,8**	**10,5**	**15,5**	**33,2**	**9,3**	**54,4**	**18,5**	**10,5**	**13,4**	**15,9**	**15,7**	**17,8**	**20,0**	**25,5**
SyntheticControl	94,7	-5,2	-4,8	-5,1	-4,9	-4,5	-4,8	-5,2	-4,0	-1,8	-0,4	-2,0	**26,9**	-4,5	-2,0
UMD	95,8	-2,5	-0,9	-2,5	-2,1	**2,2**	**0,6**	**0,4**	**2,6**	**1,9**	**3,7**	-0,8	**41,7**	**12,5**	**8,9**
no. wins		8	7	8	8	12	12	8	12	10	10	9	13	11	14

tween our approach and the referenced algorithm (positive values in the bold point advantage of our classifier, negative values indicate advantage of the referenced method). The last row contains the total number of times that our method outperformed others, providing the best possible accuracy. In direct comparison with selected algorithms, proposed approach was outperformed the most by the *HIVE-COTE v1.0* [16] algorithm, which was total of 9 out of 16 times. We were able to beat Catch22 [17] on 14 datasets which was our most favorable side by side comparison. We have turned out matching the number of the winning performances with these 4 methods: *TS-CHIEF* [15], *ROCKET* [18], *InceptionTime* [19], *ProximityForest* [20].

5. Conclusions

The main goal of our research was to propose the novel time series to image transformation and provide the way of selecting the best possible hyper parameters of this model. The output visual representation was then fed into the convolutional neural network classifier. In contrast to literature presented in Section 2, we have not put our focus into creating and optimizing multiple different network architectures but into generating the most favorable image representations of the raw series. Our method was mainly evaluated on multi class univariate samples with length varying from 20 to 3000. Conducted series of experiments shows differentiated performance depending on the evaluation set. There are multiple data collections like *Chinatown, Coffee, Herring, Meat, Rock* where our method outperformed many different state-of-the-art algorithms or matched their performance. There have also been the cases that our approach was not capable of providing as high results as the competitors on datasets like *DistalPhalanxOutlineAgeGroup, GunPointAgeSpan, GunPointMaleVersusFemale, SyntheticControl*. At this point it is worth mentioning that commonly available repositories does not include the universal method nor universal parameters settings that are capable of providing high performance on all mentioned datasets. This case is also present in our approach. It is not possible to preset the model with constant hyperparameter values and obtain competitive results on every dataset. Careful analysis of each sample allows the model to generate separate best performing combination specifically for individual series collection. In order to make searching process more organised, we have provided several conclusions in Section 4.1.

The future work related to this research can be focused on extending the approach in order to classify the multivariate time series samples. The analysis and study of presented type of the image generating process has not been exhausted. Many different alterations of this approach are yet to be discovered and evaluated. One disadvantage of presented method is the constraint regarding the length of generated SAX word. Choosing another way of arranging resulting squares could eliminate the problem and will also be the subject of further modifications.

References

1. K. Pratik & P. Mamta. (2020). ECG Heartbeat Arrhythmia Classification Using Time-Series Augmented Signals and Deep Learning Approach. Procedia Comp. Sci. 171.
2. A. Joseph, M. Abdallah & C. Raphaël. (2020). Using DenseNet for IoT multivariate time series classification. 1-6.
3. A. Monica & K. Ahsan. (2021). Applications of shapelet transform to time series classification of earthquake, wind and wave data. Engineering Structures. 228. 111564.
4. C. Myers, L. Rabiner, & A. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition." IEEE Transactions on Signal Processing, IEEE Transactions on, vol. 28, no. 6, pp. 623–635, 1980. [Online].
5. J. Lines et al. (2012). "A shapelet transform for time series classification." Proc. of the Intern. Conf. on Knowledge Discovery and Data Mining, ACM SIGKDD. 289-297.
6. P. Schäfer. "The boss is concerned with time series classification in the presence of noise." Data Mining and Knowledge Discovery 29(6), 1505–1530 (Nov 2015).
7. M. Middlehurst, W. Vickers & A. Bagnall. Scalable dictionary classifiers for time series classification. In: Intelligent Data Engineering and Automated Learning – IDEAL 2019, pp. 11–19. Springer International Publishing (2019).
8. P. Schäfer, U. Leser. "Fast and accurate time series classification with weasel." In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 637–646. CIKM '17, ACM, New York, NY, USA (2017).
9. Z. Wang, T. Oates. "Imaging time-series to improve classification and imputation." In: Proc. of the 24th Intern. Conf. on AI. pp. 3939–3945. IJCAI'15, AAAI Press (2015)
10. W. Homenda, A. Jastrzebska & M. Wrzesien. (2022). Time Series Classification Using Images. In R. A. Buchmann et al. (Eds.), Proceedings of the Information Systems Development Conf. (ISD2022 Proceedings). Cluj-Napoca, Romania.
11. N. Hatami, Y. Gavet & J. Debayle. "Bag of recurrence patterns representation for time-series classification. "Pattern Analysis and Applications 22(3), 877–887 (Aug 2019).
12. C. Benegui and R. T. Ionescu, "Convolutional Neural Networks for User Identification Based on Motion Sensors Represented as Images" IEEE Access, vol. 8, pp. 61255-61266, 2020
13. J. Lin et al. (2003). "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms." Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD 2003. 2-11.
14. A. Jastrzebska. (2020). Lagged encoding for image-based time series classification using convolutional neural networks. Statistical Analysis and Data Mining: The ASA Data Science Journal. 13. 10.1002/sam.11455.
15. A. Shifaz, C. Pelletier, F. Petitjean, et al. TS-CHIEF: a scalable and accurate forest algorithm for time series classification. Data Min Knowl Disc 34, 742–775 (2020).
16. J. Lines, S. Taylor and A. Bagnall, "HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles for Time Series Classification," 2016 IEEE 16th Intern. Conf. on Data Mining (ICDM), Barcelona, Spain, 2016, pp. 1041-1046
17. C.H. Lubba et al. catch22: CAnonical Time-series CHaracteristics. Data Min Knowl Disc 33, 1821–1852 (2019).
18. A. Dempster, F. Petitjean & G.I. Webb, ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min Knowl Disc 34, 1454–1495 (2020). https://doi.org/10.1007/s10618-020-00701-z
19. H. Ismail Fawaz, B. Lucas, G. Forestier, et al. InceptionTime: Finding AlexNet for time series classification. Data Min Knowl Disc 34, 1936–1962 (2020).
20. B. Lucas, A. Shifaz, C. Pelletier, et al. Proximity Forest: an effective and scalable distance-based classifier for time series. Data Min Knowl Disc 33, 607–635 (2019).