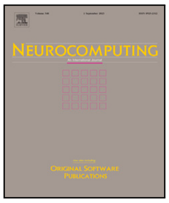
Neurocomputing 573 (2024) 127214

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Time series classification with their image representation

Władysław Homenda [a,b,*], Agnieszka Jastrzębska [a], Witold Pedrycz [c,d,e], Mariusz Wrzesień [b]

[a] The Faculty of Mathematics and Information Science, The Warsaw University of Technology, Warsaw, Poland
[b] The Faculty of Applied Information Technology, The University of Information Technology and Management, Rzeszów, Poland
[c] The Department of Electrical and Computer Engineering, The University of Alberta, Edmonton, AB T6G 2R3, Canada
[d] The Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
[e] The Faculty of Engineering and Natural Sciences, Department of Computer Engineering, Istinye University, Sariyer/Istanbul, Turkiye

ARTICLE INFO

Communicated by Q. Wen

Keywords:
Time series
Data to image transformations
Classification
Convolutional neural networks

ABSTRACT

The study is concerned with the problem of classification of multivariate time series using convolutional neural networks (CNNs). As CNNs regard inputs in the form of images, an original image-like format of temporal data is proposed. Along this line, several design alternatives are studied by forming images with the two corresponding coordinates built by the original temporal data and their differences and second differences. An overall design process is presented with a focus on investigating time series-image transformations. Experimental studies involving publicly available data sets are reported, along with a slew of comparative analyses.

1. Introduction

The problem of pattern recognition is a classic machine learning problem. In this study, we deal with a relatively new field of pattern recognition: time series classification. The issue of classifying time series has been the subject of intense research for over a decade now. However, due to the complex structure of time series, there is still a demand for new algorithms. Furthermore, time series that need classification are generated in a vast variety of real-world domains. Each domain has its specific requirements, such as the ability to handle multi-variable data, unequal lengths.

This work is a contribution to the field of time series classification. The proposed approach transforms time series into multi-dimensional images, which are then classified using convolutional neural networks. Multivariate time series with different characteristics are studied in this work. Several selected methods of time series transformation are considered, taking into account the original series values, value changes (first differences), and changes in value changes (second differences). While an initial idea was presented by Homenda et al. [1], it should be emphasized that the issue of transformation in the approach used in this paper required detailed research. It must also be stressed that this paper, in contrast to our previous study, is focused on multivariate time series.

Therefore, this paper brings forward a detailed analysis of various time-series-to-image transformations and their impact on classification. The novelties addressed in this paper are several different transformation methods, which were not mentioned in our previous study. This paper offers a deeper insight into the time series classification pipeline based on a broader range of image-based representations. What is more, we study the relation between classifier and representation.

The motivation to work with alternative time series representation comes from a range of interesting results reported in the literature on time series classification. Regardless the nature of the final classifier used, authors report that various representations may allow achieving higher accuracy. This could be explained by the fact that transformations have the ability to change statistical properties of the data, for example, differencing can help stabilize time series. Example of authors elaborating on such techniques include Gorecki and Luczak [2] (derivatives), Hatami et al. [3] (recurrence-based transformations) and others. We pair this concept with the very-popular Convolutional Neural Networks.

Other researchers also leverage graphical representation of time series as an input to CNN in order to further transform the data or perform the classification process. One example of time series imaging technique tied with the use of CNN is described in detail by authors of [4]. In one of the preprocessing steps, multi-modal electroencephalogram signals, relevant in sleep stage classification are being represented by 2D topographic mapping, which is then used as an input to custom CNN. Although the exact transformation method differs from our approach, the general idea of time series imaging remains similar.

The proposed approach is tested in an extensive empirical study, in which we compare our results with state-of-the-art methods. Our toughest competitor turned out to be WEASEL+MUSE, which provides higher

* Corresponding author at: The Faculty of Mathematics and Information Science, The Warsaw University of Technology, Warsaw, Poland.
E-mail address: wladyslaw.homenda@pw.edu.pl (W. Homenda).

https://doi.org/10.1016/j.neucom.2023.127214
Received 6 April 2023; Received in revised form 18 October 2023; Accepted 27 December 2023
Available online 30 December 2023

average accuracy across a range of datasets. However, WEASEL+MUSE is very memory-intensive. In addition, if we count the number of wins instead of average accuracy, the proposed best configurations slightly outperforms it.

This work is structured as follows. Section 1 outlines a general look at the problem of classifying time series. Section 2 is devoted to an overview of the essential works in this field. Section 3.1 describes convolutional neural networks and their application to this study. Section 4 is central to this study. They describe transformations of time series into two-dimensional images. Section 5 is devoted to an empirical analysis of the method used in this work, including the results obtained. Section 6 contains a brief summary of this work and suggestions for further studies.

2. Literature review

There are many algorithms for time series classification. We can group them into several categories. A prominent one is based on distances. Methods from this group aim to evaluate the distance between time series. Studies in this area not only but also focus on experiments with different distance measures [5]. In a typical processing scenario, time series are compared pairwise to conduct the classification step after performing the distance evaluation. In this role, the nearest neighbor method is often applied.

As the second category of methods, we may give feature-based ones. In this case, the classification step is done based on data present in the form of feature vectors. Such data representation can be processed using a conventional classifier such as a neural network or a decision tree. This is an extensive group of approaches since there is a wide range of different time series feature extraction methods. We can find feature extraction domains rooted in the frequency domain in this group. For example, methods based on spectral analysis, such as discrete Fourier transform [6], discrete wavelet transform [7] or shapelet transforms [8], can be mentioned here. Features describing time series can be also based on various statistical measures such as segment-based mean or standard deviation [9].

The feature-based methods are also rich with approaches that turn the data into sequences of symbols, then partition the series into words and proceed with time series as with text data that needs to be classified. In this family of approaches, we can find methods named BOSS [10], Contractable BOSS [11], and WEASEL [12].

Time series classification methods which we further use as a benchmark during the experimental analysis can be further divided into another 3 groups. Linear, deep learning and ensemble classifiers. ROCKET classifier achieves accuracy on the level of state-of-the-art approaches while only taking a fraction of the computational time required by them. This method transforms time series samples using random convolutional kernels and further uses it to train a linear classifier. Another method worth mentioning is WEASEL+MUSE which excels at multivariate time series classification. It generates feature vectors which store relations between different dimensions of input sample. Obtained vectors are further analyzed with machine learning classifiers which output final classification result. With increasing number of more and more complicated ML and DL approaches, mtSS-SEQL+LR shows that high accuracy can be also achieved with linear classifiers. Biggest advantage of this approach over other state-of-the-art methods is the high interpretability obtained classification results.

Despite interpretability concerns, many researchers adapt popular deep neural networks to time series, as the accuracy and scalability of these approaches is exceptional. Example network architectures used for time series classification are Inception Time (IT) [13] and ResNet [14]. Another state-of-the-art method, TAPNET [15] takes advantages from both traditional methods and deep learning, by leveraging attentional prototype networks.

Last group of classifiers worth mentioning are ensemble models. These approaches take advantage of combining results from multiple different machine learning models in order to obtain the final prediction. One of benchmark ensemble classifiers is time series forest (TSF) [16]. Authors of this method proposed tree ensemble model which randomly sampled features at each tree node. TSF takes advantage of parallel computing, therefore the computation time is reduced. Another powerful ensemble classifier we would like to highlight is Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [17]. It achieves top accuracies on both univariate and multivariate datasets thanks to the variety of its building blocks. Ensemble models in HIVE-COTE are formed from classifiers of multiple domains, including phase-independent shapelets, bag-of-words based dictionaries and phase-dependent intervals. One of the components of HIVE-COTE is Random Interval Spectral Ensemble (RISE) [18]. This method is itself another tree ensemble classifier. In this method each tree is built on a distinct set of Fourier, autocorrelation and partial autocorrelation features. RISE excels at time series classification although its run time complexity is really high which is oftentimes noticeable when training set samples are of greater length.

3. Introductory remarks

Convolutional Neural Networks are widely used in many different research areas and have also proved suitable in our approach. Further sections describe the vast amount of CNN applications, explain the meaning of individual building blocks of this network, and present selected architectures of the CNNs used in our research.

3.1. Convolutional neural networks in different domains of research — an overview

The high performance of convolutional neural network models made them highly popular in many practical domains. One example could be medical image classification. Esteva et al. [19] discussed the classification of skin lesions using a single CNN. Yadav and Jadhav [20] applied convolutional neural networks to classify pneumonia using chest X-ray images. Face recognition [21] and natural language processing tasks such as sentence classification [22] or sentence modeling [23] are also obtained with the use of CNNs. Even time series forecasting could benefit from a predictive model constructed with CNNs [24,25]. Regarding the topic directly related to this article, CNNs were also previously used for time series classification problems [26], and results achieved with this approach outperformed many state-of-the-art methods.

Convolutional neural networks (CNNs) are deep learning models mostly used in image processing tasks. It could be said that they aim to mimic the vision processes that our brains perform. The architecture of CNNs was highly inspired by the neurons of cats' visual cortices [27]. It was further modified and improved by Fukushima [28], who introduced the approach consisting of two new layers: the convolutional and downsampling layers. These layers are responsible for so-called feature extraction. Necessary transformation, which is constantly used in modern CNNs — max pooling, was introduced by Yamaguchi [29]. This concept is used as the downsampling layer. A set of mathematical operations mentioned above (described in the following sections) is applied to specific image parts to achieve the desired classification results. One of the most significant advantages of CNNs is that they can find a particular feature of an image regardless of its position. The learning process of CNN is based on fine-tuning the weights of the so-called filter. At first, these values were picked by hand, but later, some algorithms, such as back-propagation [30], were used for this problem. This approach helped achieve better performance and made CNN classifiers fit better for larger images.

3.2. *Building blocks of convolutional neural networks*

Typical CNN architectures can be divided into two parts: a feature extraction part and a classifying part. The feature extraction part stacks several convolutional layers (each one generally followed by an activation function), then a pooling layer, then a few convolutional layers, then a pooling layer, and so on. The model takes as an input an image, which is a multi-dimensional rectangular matrix of real numbers. The number of dimensions corresponds to the number of color channels. The input is transformed regarding its dimensionality and values stored in the matrix using the operations defined for each neural layer. After several transformations using convolutional and pooling layers, the signal is passed to the classifying part of the CNN. The final layer outputs the prediction (for instance, it would be a softmax layer that outputs estimated class probabilities).

In convolutional layers, we employ convolution operation with the use of filters. When broken down into elementary operations, a convolutional filter moves along the image and performs operations that resolve to multiplications and summations. In contrast, pooling is responsible for both the downsampling of an image and extracting more specific features. Similarly to convolutional layers, here, the concept of a traversing window is also present. In pooling, the two most used operations are maximum and average. For each window position, only one value is returned, and the window size determines the level of downsampling.

The next building block of CNNs is a fully connected dense layer. It ensures that connections exist between every node of the two considered layers. These layers are responsible for the final classification process and may use different activation functions. These functions define the relation between the input and output values from the neuron. Choosing the proper activation function could have a direct impact on the achieved performance of the model. It is not predetermined that the same function must be used in the entire network. Often, different types of activation functions are used for hidden layers and another one for output layers. Generally, the most commonly used functions are ReLU (rectified linear unit), Sigmoid (logistic function), and Softmax. ReLU is efficient in implementation, and it is effective. Therefore, it is often an excellent choice for hidden network layers. Sigmoid returns values from interval $[0, 1]$, so it helps present probabilities. It could be used, for instance, as the activation function for the output layer of a binary classification model.

Softmax also returns values from $[0, 1]$ interval, but they are arranged to form the vector of probabilities. This feature of the softmax function makes it a reasonable choice for the output layer of multi-class classification model. Using that method, we can obtain the information that tells the probability of the sample belonging to a specific class.

The dropout regularization method, one of the simplest ways to prevent overfitting in neural networks, is used in dropout layers. Its job is randomly excluding a defined percentage of nodes in a specific layer. Excluding nodes means that weights calculated by them will not be taken under consideration while optimizing the model. This technique vastly decreases the probability of the network suffering from overfitting and helps to obtain better classification results.

All in all, a typical CNN architecture comprises of the following layers and operations:

(A) convolutional layer;
(B) activation functions;
(C) pooling layer;
(D) dropout layer;
(E) flattening;
(F) fully-connected layer;
(G) output layer;

Layers and operations A–D are usually repeated multiple times with different internal parameters (like window size). It is also possible to apply multiple convolution/activation operations in a row (for example, making a chain AB-AB-AB-C.... All these operations are deemed to extract features from an image that is being passed through the network. Layers and operations E–G are like these in a standard feed-forward neural network and are responsible for the classification step.

3.3. *Convolutional neural network architectures: the proposed approach*

In the experiments presented in this paper, we have created and evaluated four different convolutional neural networks for transformations and their variants, cf. Section 5.1. The hyperparameters of these models are presented in Table 1 — many combinations of parameters were tested to find models that can achieve satisfactory results. In Table 1, we separate the information concerning a feature extraction part from the information relating to a classifying part of each CNN. Regarding the first part, models differed in the number of convolutional and pooling layers, the size of kernels used, and the number of neurons in each layer. Classification parts of created neural networks used different numbers and parameters of dropout layers. Still, all models performed final classification using the softmax activation function in the output layer.

A summary of the architectures of the CNN models used in the experiments is given in Table 1. We have worked on relatively shallow models with clearly outlined feature extraction and classification parts.

As an example, a more detailed description of Model #1 will be given in this paragraph. To obtain the model, we used all layers mentioned in the previous section. It uses two sets of convolutional and max pooling layers followed by a few fully connected layers with ReLU as the activation function. Categorical cross-entropy was used as a metric for calculating the loss during the training of our model. The last layer of the model was set to use the softmax activation function. Initial hyperparameters of the network were proposed, but they could have been slightly changed based on the model's performance on the test set.

We used two dropout layers with a probability coefficient of 0.15, to prevent our network from overfitting. As mentioned, this layer excludes random input values to layers (in our case, 15% of them).

Thanks to the combination of categorical cross-entropy as a loss function and softmax as an activation function, we could have expected high quality of achieved results. For each image, the vector of probabilities is returned with a size equal to the number of classes in the dataset. The index of the highest probability value represents the prediction of the class the image belongs to.

4. Time series transformations — methodology of research

The method presented in this paper enables the classification of time series regardless of their nature. The procedure allows the classification of both one-dimensional and multi-dimensional (multivariate), binary, and multi-category datasets. The approach transforms numerical time series into images and then employs a convolutional neural network model (CNN) described in the previous paragraph to classify the data in this form. The essential and partly new solution is the prior transformation of the time series into a two-dimensional image of the size $w \times h$ ($width \times height$). As a rule of thumb, we recommend $w = 400$ and $h = 400$. These values are used in the experimental part of this paper. If the time series is described by multiple variables (dimensions), a separate image channel would be generated for each of these d variables, as illustrated in Fig. 1.

Fig. 1 presents a single three-variable time series from the Character Trajectories dataset. To process multi-variate time series using the proposed strategy, one generates an image for each variable separately and passes it together to the input of a CNN to recognize it. The images corresponding to subsequent variables in the time series are treated as separate image channels. We envision this transformation on the conceptual level, as presented in Fig. 2. Particular values, denoted in

Table 1
Summary of the architectures of the CNN models.

	Model #1	Model #2	Model #3	Model #4
Feature-extraction part of the CNN		Convolution window: (3 × 3) Number of output filters: 64 MaxPooling with window (2 × 2) ReLU	Convolution window: (1 × 3) Number of output filters: 64 MaxPooling with window (1 × 2) ReLU	Convolution window: (1 × 3) Number of output filters: 64 MaxPooling with window (1 × 2) ReLU
	Convolution window: (5 × 5) Number of output filters: 32 MaxPooling with window (2 × 2) ReLU	Convolution window: (3 × 3) Number of output filters: 64 MaxPooling with window (2 × 2) ReLU	Convolution window: (1 × 3) Number of output filters: 64 MaxPooling with window (1 × 2) ReLU	Convolution window: (1 × 3) Number of output filters: 64 MaxPooling with window (1 × 2) ReLU
	Convolution window: (3 × 3) Number of output filters: 32 MaxPooling with window (2 × 2) ReLU	Convolution window: (3 × 3) Number of output filters: 128 MaxPooling with window (2 × 2) ReLU	Convolution window: (1 × 3) Number of output filters: 128 MaxPooling with window (1 × 2) ReLU	Convolution window: (1 × 3) Number of output filters: 128 MaxPooling with window (1 × 2) ReLU
Classifying part of the CNN	192 units, ReLU	256 units, ReLU	256 units, ReLU	256 units, ReLU
	Dropout (0.15)	Dropout (0.1)	Dropout (0.1)	Class_num, softmax
	128 units, ReLU	Class_num, softmax	Class_num, softmax	
	Dropout (0.15)			
	Class_num, softmax			

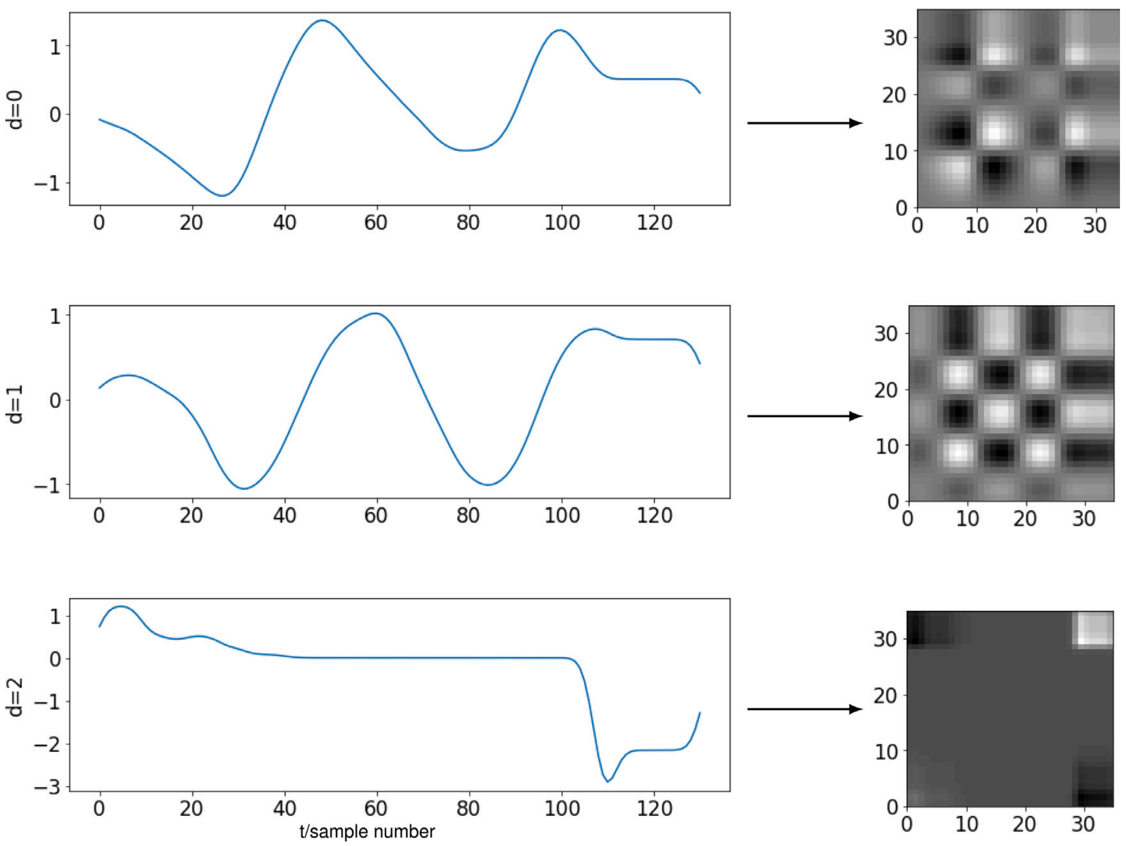


Fig. 1. An illustrative example of time series transformation for $d = 3$ into images using the proposed approach, cf. [31].

this figure with symbols $x_{i,j}$, $y_{i,j}$, and $z_{i,j}$ are computed with the use of the proposed time series transformations.

After analyzing many possible transformations, we decided to present three selected ones. The first transformation has two variants,

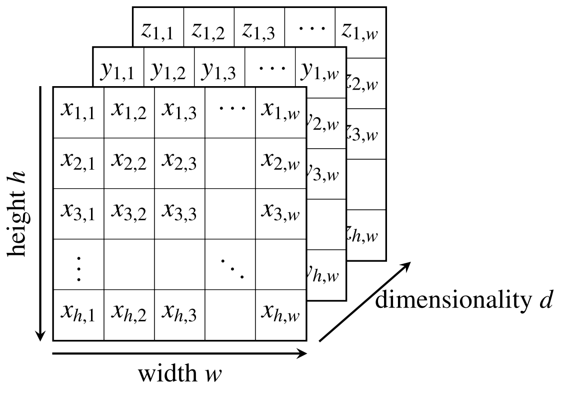

$z_{1,1}$ $z_{1,2}$ $z_{1,3}$ ⋯ $z_{1,w}$
$y_{1,1}$ $y_{1,2}$ $y_{1,3}$ ⋯ $y_{1,w}$
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ ⋯ $x_{1,w}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,w}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{3,w}$
$x_{h,1}$ $x_{h,2}$ $x_{h,3}$ $x_{h,w}$
height h
width w
dimensionality d

Fig. 2. Generated data structure.

presented in Section 4.1. The second transformation is described in Section 4.2. Two variants of the third transformation are outlined in Section 4.3. Section 4.4 covers an illustrative case study in which we employ these transformations. The case study uses the Character Trajectories dataset introduced in [31].

4.1. Transformation #1: Values, value changes, changes of value changes

The two variants of these transformations are based on original time series data, differences, and second differences of original data. The global parameters for both the presented variants are the width w and the height h of the generated images and the color scale count c, which will be responsible for the color contrast in the image. In addition, the parameter d, the number of images, is determined automatically and depends on the number of variables (dimensions) of a given series.

The following preprocessing is performed for each dimension (out of d) of the given time series based on primary data, i.e., based on source time series stored in an array of length n:

- time series values: $a_1, a_2, a_3, a_4 \dots, a_n$, value changes: $da_2, da_3, da_4 \dots, da_n$, and changes of value changes: $dda_3, dda_4, \dots, dda_n$, that is, the original values, the first and the second differences, are calculated:
  - the first difference is the difference of consecutive values of the series: $da_k = a_k - a_{k-1}$ for $k = 2, 3, \dots, n$,
  - the second difference is the difference of values of consecutive first differences: $dda_k = da_k - da_{k-1} = a_k - 2 * a_{k-1} - a_{k-2}$ for $k = 3, 4, \dots, n$,
  - these values are then stored in arrays of the same size as the input series except, of course, the first one or the first two elements of the arrays,
- the following sequences of data pairs are considered for the two variants of this transformation:
  - values and value changes, i.e. (a_k, da_k) where $k = 2, 3, 4, \dots, n$,
  - value changes and changes of value changes, i.e. (da_k, dda_k) where $k = 3, 4, \dots, n$,
- the preprocessed values are converted into d images (two-dimensional arrays of pixels) for each sequence of pairs, with assumed width w and height h, i.e., images of size $w \times h$,
  - the pixels of the images (elements of the arrays) are initiated with value 0,
  - the original values and differences are re-scaled to the integer values from the intervals $[0, w]$ and $[0, h]$ for the first variant. Similarly, differences and second differences are re-scaled to the intervals $[0, w]$ and $[0, h]$ for the second variant. Let denote by i_k and j_k rounded values of either a_k and da_k or da_k and dda_k, respectively,
  - for each pair i_k and j_k, $k = 3, 4, \dots, n$, the (i_k, j_k) pixel of the corresponding image (the cell of the corresponding table of size $w \times h$) is incremented by 1 until its value is less than c.

In the above process, two images are produced. Let us call them Val/ValChng and ValChng/ChngValChng.

Notice that when the value (grayscale count) of the given pixel of the generated image reaches the defined maximum value c, then the value of this pixel is not increased anymore. This restriction comes from observations made in preliminary studies that a large portion of the pairs of values of the first and second differential are close to the point $(0, 0)$. This effect significantly increases the grayscale (or color) values, negatively affecting the ability to differentiate images. Also, observations of tests concluded that no real benefit could be gained with the pair of sequences Val/ChngValChng.

Finally, two images are produced: Val/ValChng and ValChng/ChngValChng. They are arrays of dimensions $w \times h \times d$ filled with integer values from the interval $[0, c]$, which are then passed as input to a neural network-based classifier.

4.2. Transformation #2: values × values

In contrast to variants Val/ValChng and ValChng/ChngValChng, only the actual values of the examined time series, $a_1, a_2, a_3, \dots, a_n$, are considered in the preprocessing and image generation. The time series values are scaled to the unit interval $[0, 1]$ employing the transformation $b_k = (a_k - min)/(max - min)$, where $k = 1, 2, 3, \dots, n$ and min and max are minimal and maximal values of the original values of the time series. The scaled values are interpreted as grayscale values, with 0 representing *black* and 1 representing *white*. The scaled time series can be seen as a grayscale image of size $n \times 1$.

Then, an $n \times n$-pixel image is created. The pixel $[i, j]$ is filled in with the product of the ith and jth element of the re-scaled series, i.e., is filled in with $b_i * b_j$. As in the previous variants, a separate image is created for each dimension. The result gives an array of dimensions $n \times n \times d$, containing the calculated values.

4.3. Transformation #3: Replicated series of values and of value changes

Based on the preliminary experiments, we decided to prepare images better suited to the learning algorithm used, i.e., to the convolutional networks. The approach also brings advantages due to the great possibilities in applying filters. For example, the difference operation would correspond to a convolution with a filter $[-1, 1]$.

There are two variants to this approach. The image generation in the first variant is based on the source data without performing any further transformations: $a_1, a_2, a_3, \dots, a_n$. The image prepared will be of size $n \times 1$, where n is the length of the series, i.e., it is a vector of values. Then, this vector is replicated to create an image of size $n \times n$. Thus, the images will consist of vertical bars whose color/shade of gray will correspond to the (scaled) value of the series at a given time point.

The second variant is like the first one, except that the first derivative $da_2, da_3, \dots, da_n$ replace original values. Let us call the above two variants ReplVal and ReplValChng.

4.4. Illustrative case study of the proposed transformations

In order to further explain the proposed methods that transform time series into images, the result of the mentioned methods will be illustrated with example images that were generated for the publicly available CharacterTrajectories dataset. It is the same dataset that we used in the descriptive part of our previous paper on image-based representations for time series classification. It contains 2858 character

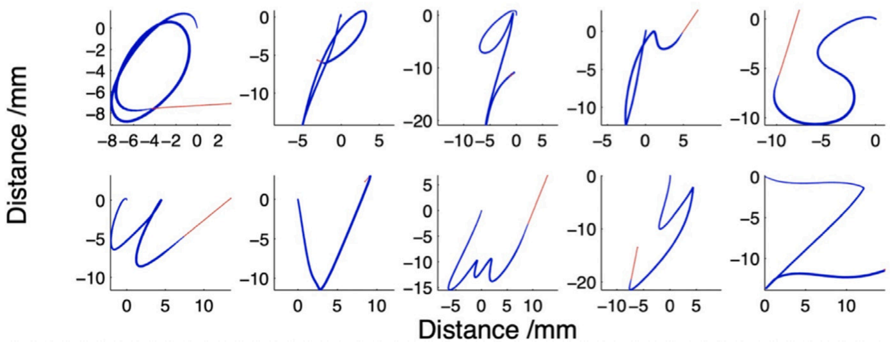


Fig. 3. 10 samples of characters taken from the mixed dataset. The total size of the dataset is 2858 characters, with over 100 samples of each character, cf. [31].

samples that were collected using a tablet and then normalized. Each image is described by three dimensions x, y, and pen tip force. The class label is one of 20 characters ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘g’, ‘h’, ‘l’, ‘m’, ‘n’, ‘o’, ‘p’, ‘q’, ‘r’, ‘s’, ‘u’, ‘v’, ‘w’, ‘y’, ‘z’. Examples of characters are displayed in Fig. 3. In order to standardize the base, all series have been shortened to the shortest length, 182. Shortening may affect the classification, but we do not care since it is used here only for illustration.

The time series in the discussed set has three dimensions. Therefore, three images will be generated for each proposed variant of transformations into an image for the selected set. In the subsequent Figures, the left image illustrates the first dimension $d = 1$, i.e., it illustrates the x variable of the coordinate system. The middle image is for dimension $d = 2$, i.e., it represents the y variable. The right-most image depicts the third dimension $d = 3$ that exemplifies the pen force. In order to illustrate the differences between the classes for each variant, images for two different classes will be presented. Since the purpose of this paper is not only to process the problem outlined in [31], the presented images will not be analyzed in terms of specific classes, i.e., in terms of written characters — instead, the discussion concerns the performance of the proposed data transformations in general. Also, the top images are for one class in the subsequent Figures, and the bottom images are for another class.

4.4.1. Transformation #1 & #1.1: Values, value changes, changes of value changes

In variant Val/ValChng, the points on the image represent the actual values of the time series (position) and the value of the first derivative (velocity) — Fig. 4. Due to the specificity of the set for which sample images were prepared, the data were standardized in the range $[-3, 3]$. As an example for two different classes, the prepared images for each class have characteristic points that the neural network-based classifier should notice.

For the variant ValChng/ChngValChng, the generated points visibly arrange themselves into clusters regardless of class and dimension. In most cases, the generated images seem to be similar. However, after a closer analysis, fundamental differences between the classes are visible, which is presented in Fig. 5. This observation suggests that neural network learning to distinguish classes based on small differences should “see” these differences.

4.4.2. Transformation #2: Values × values

In the variant Values x Values there are no more pairs of points, only Cartesian product of the transformed actual values of the series by themselves. The resulting images have a very characteristic structure, similar to a chessboard, and are symmetric concerning the diagonal of Fig. 6.

In addition, when the values are scaled to the interval $[0, 1]$, most pixels have a value close to zero (black), and some areas have a value close to 1 (white). The distribution can distinguish the classes of the series, the number of resulting squares in the image, and the appearance of these areas.

4.4.3. Transformation #3 & #4: Replicated series of values and value changes

Before analyzing the images for replicated series of values and series of value change variants, recall that the height of the source image is 1. Hence, vertical bars are generated in a square plot by replicating that one-row image. The variant of the repeated series of values shows many differences between the images of different classes in Fig. 7. Analyzing the images for the bottom dimension and the different classes more closely shows many differences in the created striped images. The width of the stripes and the gradients in the transitions between the light and dark areas differentiate classes of images. Therefore, it can be assumed the classifiers probably could learn the differences between the classes on this basis.

In the variant replicated series of value changes, the images are generated similarly to the variant replicated series of values. However, instead of the actual values of the series, the differences in subsequent values are taken. Images in this variant are similar to images from the previous variant, although differences between classes seem to be more visible in Fig. 8.

4.5. Final remarks

The method handles well datasets containing missing values. The simplest solution employed in this study is to discard the missing values while computing the transformations. When the time series is multidimensional, and there is a missing value $x_{k,j}$ in any dimension j, values from the remaining dimensions $x_{i,j}$ should also be discarded for each $i = k$, where i is the number of measurement, and j is the number of dimension.

It is also good to underline that by design, a CNN explores the relationships between adjacent pixels. That is, the order of pixels must matter, and swapping pixel columns or rows by place will result in entirely different images. For each of the proposed variants – for example, for variant ValChng/ChngValChng – each pixel position is related to the first and second derivative of the discussed series. In contrast, in the other variants, the position of pixels in the image is determined by the order of measurements.

The transformations allow the preparation of 5 different images for a given time series for one-dimensional sequences and $5 \times d$ images for d-dimensional data.

5. Experimental studies

The experiments were performed for all five transformations described in the previous sections. The proposed custom classifier was based on neural networks.

The experiments were designed and implemented using the Python programming language. The data for the experiments was taken from the public repository available under . The basic characteristics of datasets are given in Table 2. The choice of the datasets was patterned on the selection made by Ruiz et al. [32], who have recently published a comprehensive survey on multivariate time series classification. In this study, we consider exactly the same datasets.

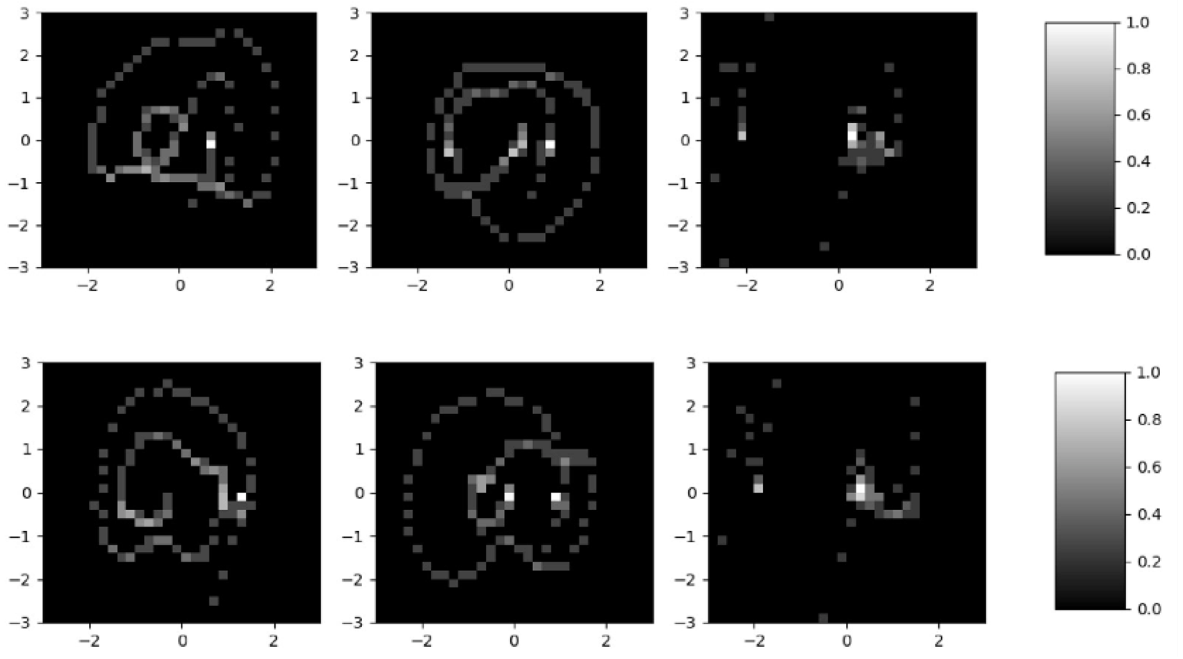


Fig. 4. Transformation #1: Data processing using Val/ValChng transformation.

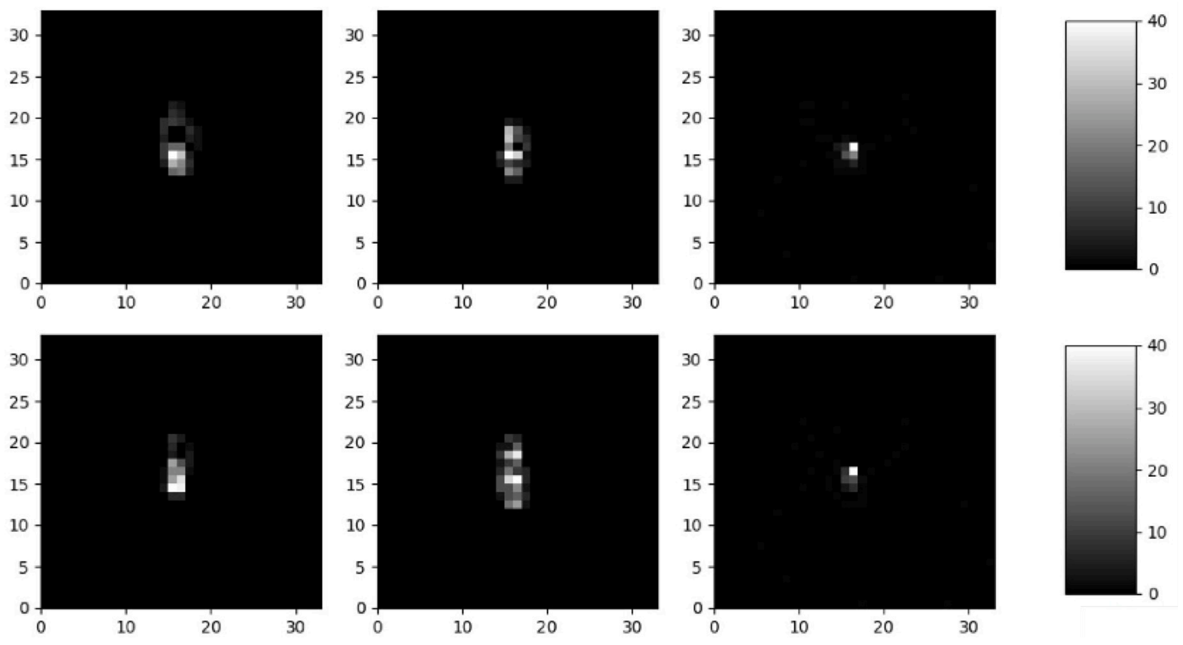


Fig. 5. Transformation #1.1: Data processing using ValChng/ChngValChng transformation.

5.1. *Experiments*

In line with the proposed approach outlined in the previous paragraphs, tests were carried out for the time series described in Table 2. Table 1 summarizes the CNN model architectures recommended for individual variants of data preprocessing described in Section 3.1.

As a part of the study addressed in this paper, we have performed cross-testing of various representations and CNN architectures. We have tested all representation/CNN architecture combinations mentioned in 3 and 4. This study aimed to propose a ready solution that “usually” provides good results. As an outcome of these experiments, CNN models were paired with representations. It was decided to conduct experiments on relatively shallow CNN models in which the layers related to feature extraction and classification are clearly visible. The choice of the suggested representation/CNN architecture pair was based on a sum of average accuracy for all considered datasets. Admittedly, there are cases where, for some particular dataset, swapping CNN architectures could help increase the accuracy of time series recognition. However, we wanted to deliver some precise suggestions concerning a model setup that is generally working well. Naturally, tuning of the CNN model is the most obvious point that can be done when one wishes to improve method’s performance.

The experiments were carried out as follows. Datasets were downloaded from the time series repository . The files already contain a split into train and test sets. In order to mitigate the influence of randomness on the reported values, the training procedure was repeated ten times

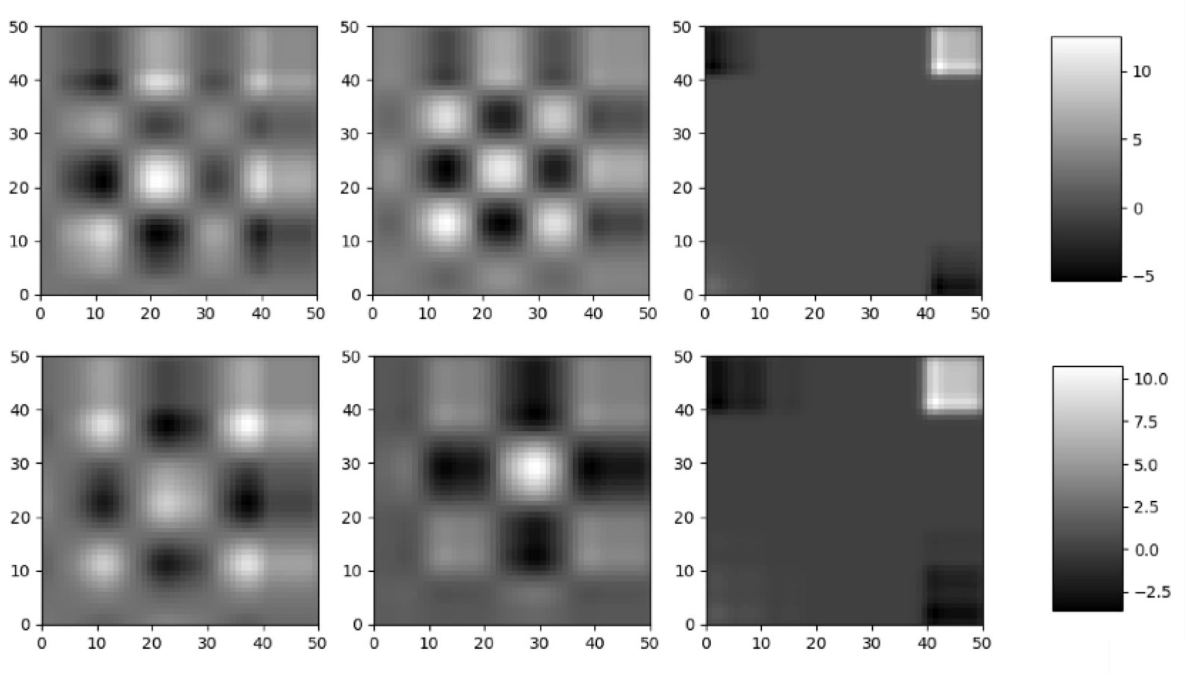


Fig. 6. Transformation #2: Data processing using Values × Values transformation.

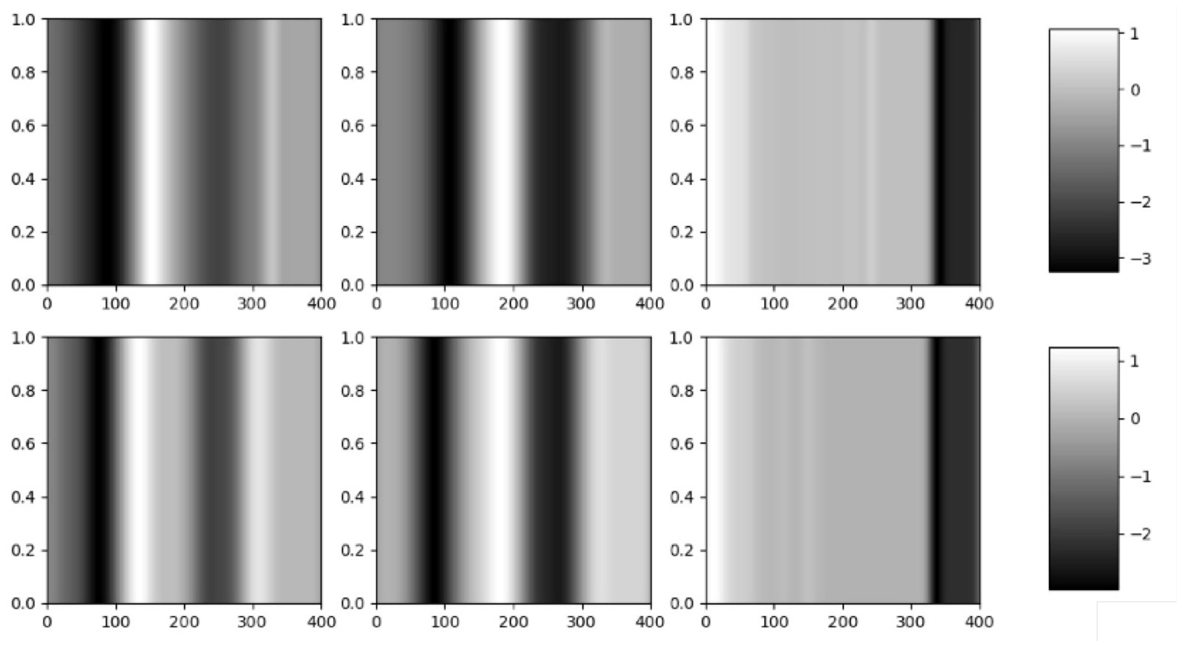


Fig. 7. Transformation #3: Data processing using replicated series of values transformation.

in each case. The training utilized a train set of a fixed size and content (reported in Table 2). Model evaluation was done for test sets. In later parts of this section, we report on the outcomes of these procedures. In each case, we give statistics concerning test set results only. We look into average accuracy, maximum accuracy, and standard deviation. Since the training sets in the aforementioned repository are generally balanced, the community working on time series classification reports quality mainly in terms of accuracy, and so do we.

5.2. Results — average accuracy

Let us compare the accuracy of time series classification based on images generated with five methods. Table 3 shows the average accuracy achieved for 26 test sets. The last row includes the average accuracy of the five methods on all 26 datasets (average for each column). The table concerns the recommended pair of transformation/CNN architecture.

After analysis of the results, it is visible that the accuracy of the ReplVal variant (Representation #3) was the best in 14 out of 26 cases. Also, the highest average accuracy for all 26 datasets was obtained by this variant (see Table 3). The second-best approach was based on Representation #4. The third place was taken by Representation #1.1. The fourth place was taken by Representation #2. In the end, we see Representation #1. The differences between Representations #1, #2, and #3 are relatively significant. It is a very interesting result.

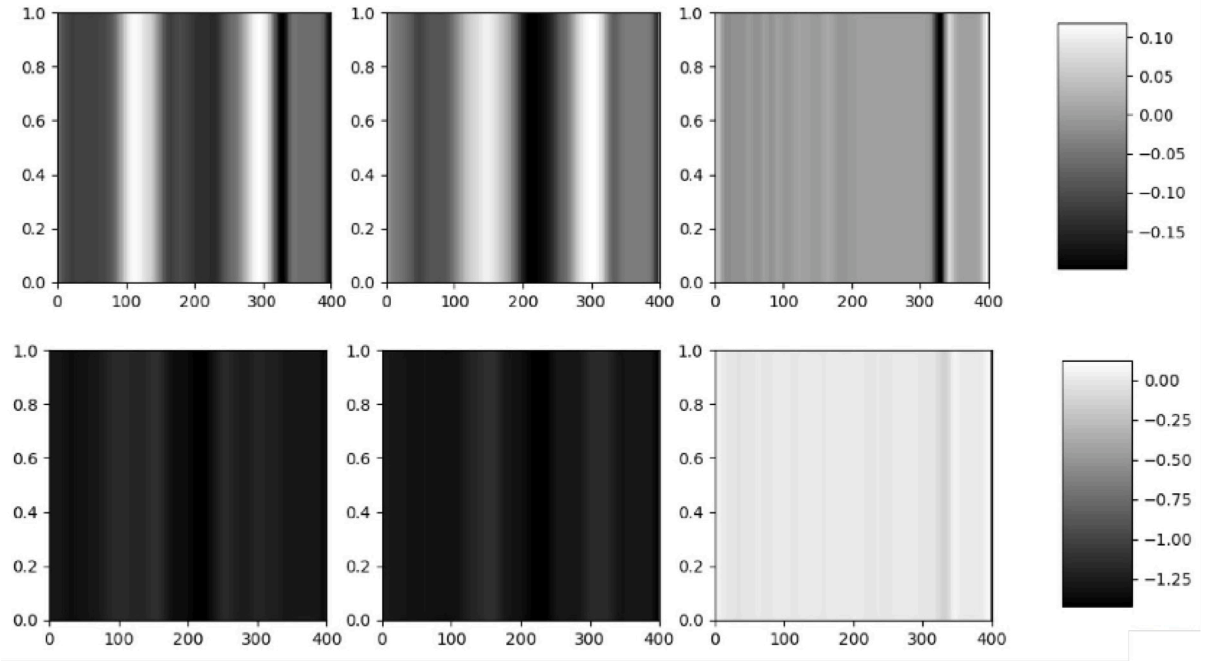


Fig. 8. Transformation #4: Data processing using replicated series of value changes transformation.

Table 2
Characteristics of selected datasets.

Dataset	Train size	Test size	Length	Number of classes	Number of dimensions
ArticularyWordRecognition	275	300	144	25	9
AtrialFibrillation	15	15	640	3	2
BasicMotions	40	40	100	4	6
Cricket	108	72	1197	12	6
DuckDuckGeese	60	40	270	5	1345
EigenWorms	131	128	17 984	5	6
Epilepsy	137	138	207	4	3
ERing	30	270	65	6	4
EthanolConcentration	261	263	1751	4	3
FaceDetection	5890	3524	62	2	144
FingerMovements	316	100	50	2	28
HandMovementDirection	160	74	400	4	10
Handwriting	150	850	152	26	3
Heartbeat	204	205	405	2	61
Libras	180	180	45	15	2
LSST	2459	2466	36	14	6
MotorImagery	278	100	3000	2	64
NATOPS	180	180	51	6	24
PEMS-SF	267	173	144	7	963
PenDigits	7494	3498	8	10	2
PhonemeSpectra	3315	3353	217	39	11
RacketSports	151	152	30	4	6
SelfRegulationSCP1	268	293	896	2	6
SelfRegulationSCP2	200	180	1152	2	7
StandWalkJump	12	15	2500	3	4
UWaveGestureLibrary	2238	2241	315	8	3

5.3. A study on the relation between CNN model and representation

Let us inspect the differences between various CNN architectures in a scenario when we fix the representation scheme. This experiment allows us to get some insights into the roles of representation and classifier in the proposed pipeline.

We have conducted an experiment in which we trained all CNN models with a given representation. We focus on Representation #1, #2, #3, and #4. We skip Representation #1.1, as it is a special case of Representation #1. The outcomes of these tests are analyzed in the form of average accuracy and average standard deviation achieved for a given pair representation/CNN model. The outcomes of this experiment are displayed in Table 4. A property of a good setup is when accuracy is high and standard deviation is low.

Table 4 shows that the most successful representation was Representation #3. It achieved high accuracy and a relatively low standard deviation, suggesting the method's stability. The worst results were achieved for Representation #4. If we compare these results with the results given in Table 3, we see that the CNN model seems to influence the variability of the results more than data representation. CNN model #3 paired with Representation #3 provided the best results, its average accuracy was the highest. Overall, the worst-performing CNN model was Model #1. It is not surprising, as it is the least complex one. The smallest standard deviation was for CNN Model #4.

Let us present more detailed results for Representation #3. This method turned out to be the most successful. We fix this representation method and give detailed results for all CNN models for each analyzed dataset. These are gathered in Table 5 — the table presents essential statistics for ten repetitions of the training procedure. We report in percent average and maximum accuracy for test sets for all considered datasets. The results show that the average accuracy for the considered combinations of representation and the four CNN models is the highest for Model #3, which was the recommended option. The differences are not very high. In each case, it is less than 5%. This representation pairs well with many CNN models.

Table 5 shows that there are differences between different representation/model configurations. Thus, it is necessary to select the best model for a given representation.

Let us emphasize that Representation #3 was selected to be paired with Model #3. Table 5 confirms that this was the right choice. In many cases, this combination gives better results than a combination of Representation #3 with other models. When we say better, we mean more robust. The standard deviation and the maximum achieved accuracy are satisfying for this combination. Just looking at the maximum accuracy will not suffice. The standard deviation of achieved accuracy in several repetitions is also an essential measure of model quality. The differences can be substantial.

A further inspection of the method's performance is given in Figs. 9, 10, 11, and 12. We display box plots concerning the accuracy of selected datasets: BasicMotions, Cricket, Heartbeat, and NATOPS. Box

Table 3
Comparison of results obtained using different image-based methods for time series classification on 26 selected data sets (Accuracy in %) Average accuracy computed based on ten repetitions of the procedure.

	Val/ValChng	ValChng/ ChngValChng	Val x Val	ReplVal	ReplValChng
ArticularyWordRecognition	78.00%	95.67%	94.67%	**97.33%**	65.00%
AtrialFibrillation	26.67%	26.67%	33.33%	**40.00%**	**40.00%**
BasicMotions	86.50%	95.00%	**100%**	**100%**	83.75%
Cricket	68.61%	**98.61%**	93.72%	93.72%	67.78%
DuckDuckGeese	54.50%	**63.00%**	40.00%	51.40%	34.00%
EigenWorms	**74.05%**	61.83%	64.89%	57.25%	67.18%
Epilepsy	94.20%	**96.38%**	94.20%	91.30%	86.96%
ERing	62.96%	73.70%	82.96%	**91.48%**	75.93%
EthanolConcentration	26.43%	27.07%	36.88%	36.12%	**38.02%**
FaceDetection	50.55%	50.47%	65.29%	**65.80%**	61.89%
FingerMovements	50.00%	53.00%	**59.00%**	54.00%	**59.00%**
HandMovementDirection	23.51%	22.70%	37.84%	**44.59%**	31.08%
Handwriting	10.01%	10.05%	20.71%	**32.52%**	20.47%
Heartbeat	64.83%	**72.68%**	70.78%	67.22%	65.37%
Libras	72.83%	74.06%	58.33%	76.00%	54.44%
LSST	39.42%	**56.08%**	30.33%	51.18%	44.85%
MotorImagery	49.00%	53.00%	48.00%	**53.00%**	51.00%
NATOPS	73.39%	78.33%	**92.78%**	**92.78%**	73.22%
PEMS-SF	49.48%	64.86%	87.86%	81.73%	**88.44%**
PenDigits	87.39%	93.57%	95.45%	96.43%	**97.48%**
PhonemeSpectra	13.87%	12.47%	**15.69%**	13.53%	15.03%
RacketSports	56.78%	56.71%	73.03%	**83.16%**	73.68%
SelfRegulationSCP1	57.00%	62.80%	67.24%	**83.96%**	79.18%
SelfRegulationSCP2	53.89%	50.00%	50.00%	**60.56%**	52.78%
StandWalkJump	33.33%	20.00%	33.33%	**38.67%**	**38.67**%
UWaveGestureLibrary	62.72%	62.50%	82.78%	**85.63%**	83.12%
Columns/methods average	54.61%	58.89%	62.66%	**66.90%**	59.55%
Wins (incl. ties)	1	5	4	**14**	6

Table 4
Average accuracy/standard deviation of accuracy (in %) computed for all 26 datasets in ten repetitions of the procedure.

	Model #1	Model #2	Model #3	Model #4	Avg. acc/std
Representation #1	54.61/2.49	53.20/2.19	52.84/2.08	51.90/2.02	53.14/**2.20**
Representation #2	60.85/2.88	62.66/2.72	63.50/2.84	58.81/2.72	61.46/2.79
Representation #3	62.17/2.35	64.94/2.36	66.90/2.02	63.55/2.37	**64.39**/2.28
Representation #4	46.10/5.31	50.25/5.40	55.32/5.29	59.55/5.05	52.81/5.26
Avg. accuracy/st.dev.	55.93/3.26	57.76/3.17	**59.64**/3.06	58.45/**3.04**	

Table 5
Statistics: average and maximum accuracy obtained in 10 repetitions of the classifier training procedure expressed in percentages. The experiment results allow us to trace how performance changes when we fix representation to Representation #3 and swap CNN models for Model #1, Model #2, Model #3, and Model #4. Part 1.

Dataset	Model #1	Model #2	Model #3	Model #4
ArticularyWordRecognition	87.37/**99.00**	96.80/98.67	**97.33**/98.67	78.50/90.00
AtrialFibrillation	42.52/53.33	43.53/60.00	40.00/53.33	**46.16**/**66.66**
BasicMotions	96.12/97.50	98.00/**100**	**100**/**100**	84.72/90.00
Cricket	86.43/88.89	94.01/**94.44**	**93.72**/**94.44**	67.11/69.44
DuckDuckGeese	54.60/60.00	52.00/57.50	51.40/60.00	**55.00**/**62.50**
EigenWorms	73.68/92.19	76.45/94.53	57.25/75.00	**80.27**/**97.66**
Epilepsy	89.77/91.30	**92.94**/93.48	91.30/**96.38**	92.21/94.20
ERing	75.20/93.70	74.93/92.96	**91.48**/**94.82**	74.79/94.44
EthanolConcentration	34.18/36.12	32.43/36.50	**36.12**/38.40	30.91/**41.83**
FaceDetection	64.51/65.35	65.29/66.12	**65.80**/**67.82**	65.59/66.86
FingerMovements	52.20/**62.00**	52.90/59.00	**54.00**/59.00	53.30/58.00
HandMovementDirection	38.65/50.00	44.86/**55.41**	44.59/51.35	**46.35**/54.05
Handwriting	32.26/34.35	33.27/35.17	32.52/33.88	**33.89**/**36.47**
Heartbeat	55.36/68.29	54.12/66.34	**67.22**/**70.24**	55.81/68.29
Libras	75.11/77.22	74.94/77.78	76.00/77.78	**76.72**/**80.56**
LSST	28.95/30.13	49.80/50.73	**51.18**/**57.75**	46.43/48.05
MotorImagery	52.40/57.00	50.70/**60.00**	53.00/57.00	**53.90**/54.00
NATOPS	**93.50**/**96.11**	91.67/94.44	92.78/94.44	92.83/94.44
PEMS-SF	83.93/**87.28**	**84.34**/86.71	81.73/86.71	82.49/86.71
PenDigits	86.26/87.14	95.70/97.00	96.43/97.26	90.55/91.51
PhonemeSpectra	12.63/13.36	13.20/14.43	**13.53**/**14.94**	13.12/13.87
RacketSports	80.20/83.55	82.11/84.87	**83.16**/84.87	82.30/**85.53**
SelfRegulationSCP1	55.05/58.36	71.40/75.43	83.96/88.40	74.64/78.84
SelfRegulationSCP2	49.50/54.44	48.56/55.00	60.56/60.00	51.72/54.44
StandWalkJump	30.67/40.00	30.00/33.33	38.67/46.67	38.67/46.67
UWaveGestureLibrary	85.34/**87.19**	84.50/86.87	**85.63**/85.62	84.28/85.62

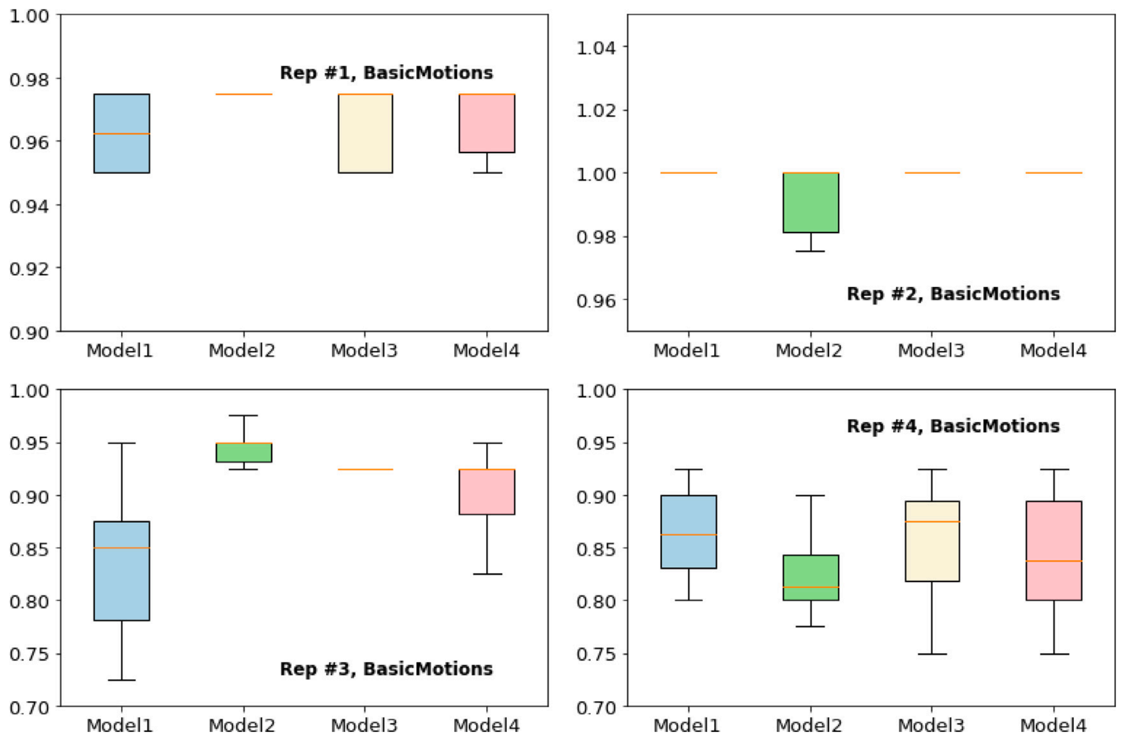


Fig. 9. BasicMotions dataset: comparison of accuracy box plots for Representations Rep #1, Rep #2, Rep #3, and Rep #4 for various CNN models (marked on the horizontal axis as Model1, Model2, Model3, and Model4.

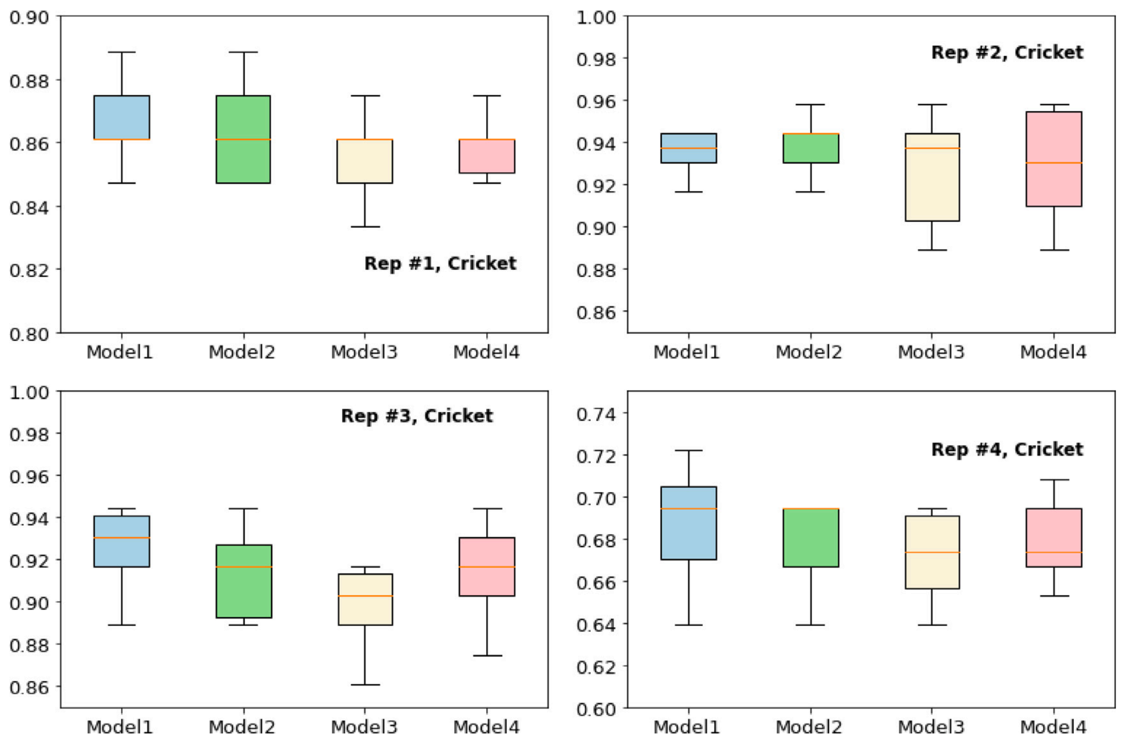


Fig. 10. Cricket dataset: comparison of accuracy box plots for Representations Rep #1, Rep #2, Rep #3, and Rep #4 for various CNN models (marked on the horizontal axis as Model1, Model2, Model3, and Model4.

plots allow us to trace in a more comprehensive manner the dispersion observed in the data. Box plots concern accuracy on test sets and are prepared for hand-picked datasets to limit the volume of the paper. In Figs. 9, 10, 11, and 12, we observe how different pairs of representation/CNN model contribute together towards correct time series recognition.

In Figs. 9, 10, 11, and 12, one shall note that there are different OY axis scales. We picked them intentionally different for each plot to focus more on a comparison for different CNN models for a fixed representation than to focus on comparisons for different representations.

For some datasets, like BasicMotions (Fig. 9), the accuracy variability observed in ten repetitions of the procedure is zero in several cases:

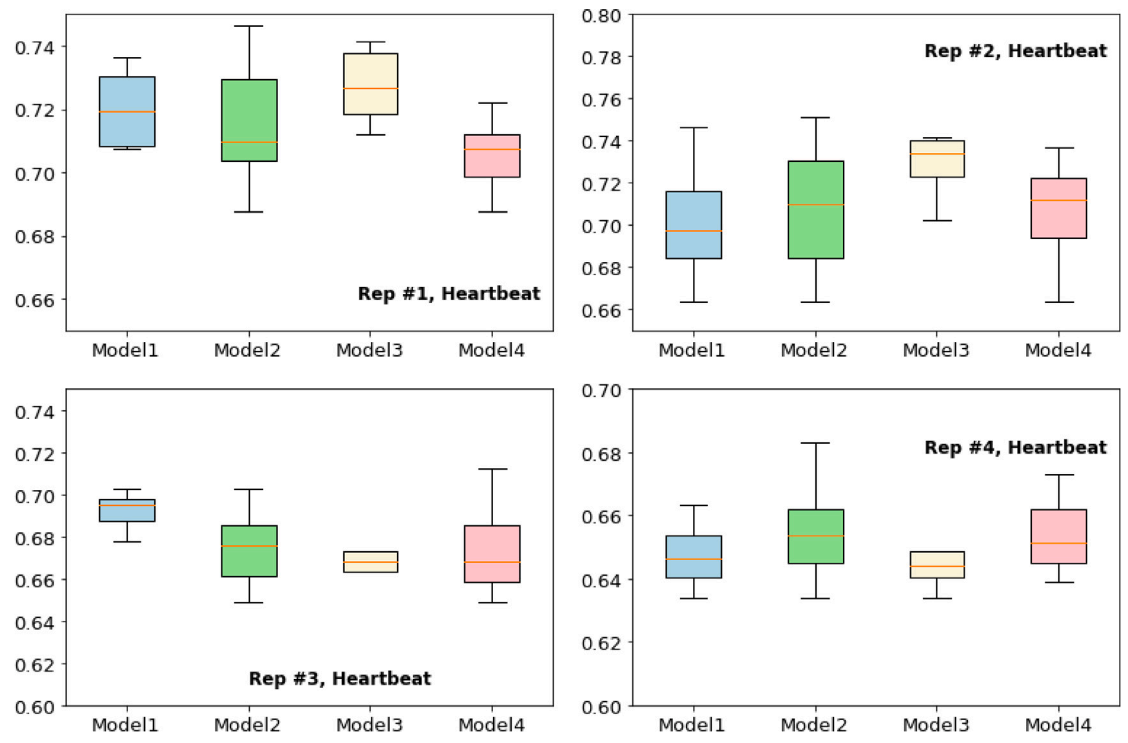


Fig. 11. Heartbeat dataset: comparison of accuracy box plots for Representations Rep #1, Rep #2, Rep #3, and Rep #4 for various CNN models (marked on the horizontal axis as Model1, Model2, Model3, and Model4.

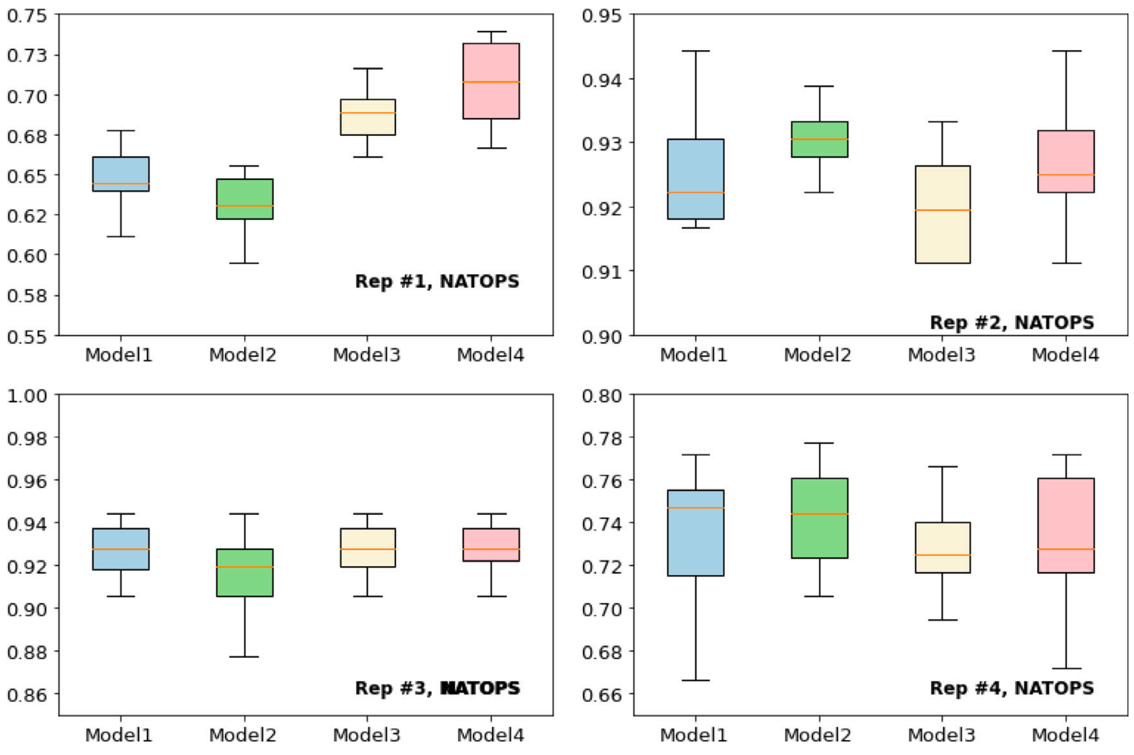


Fig. 12. NATOPS dataset: comparison of accuracy box plots for Representations Rep #1, Rep #2, Rep #3, and Rep #4 for various CNN models (marked on the horizontal axis as Model1, Model2, Model3, and Model4.

networks converge and give the same output in multiple trials. In most cases, the deviation of the accuracy is more significant. For instance, dispersion is much larger for the NATOPS dataset (Fig. 12), which is a challenging one. In an ideal situation, we would like to see high accuracy and small dispersion, which would indicate that the model is accurate and stable.

We want to state that the performed tests show that there are some “easy” datasets, such as BasicMotions, for which it is relatively easy to develop a successful and stable classifier. This dataset[1] was generated in a simple project where four subjects performed four activities while wearing a smartwatch: walking, resting, running, and badminton. Recorded are signals from the accelerometer and gyroscope. The time series exhibit differences that are easy to spot with a bare eye. In comparison, we may give the Heartbeat dataset as a challenging one[2]. This dataset concerns heart sound recordings sourced from several subjects. They were collected independently in either a clinical or nonclinical environment, from children and adults, from both healthy subjects and pathological patients, and, in addition, from different locations on the body (including but not only from the aortic area, pulmonic area, tricuspid area, and mitral area). There are two classes called normal and abnormal. The former class samples were from healthy subjects, and the latter were from subjects with confirmed cardiac diagnoses. The challenging issue is that the patients suffered from various illnesses like mitral valve prolapse, mitral regurgitation, aortic stenosis and others. Each dimension is a frequency band from the spectrogram. For this dataset, we observe relatively high standard deviations in classifiers' accuracy and the differences for different processing pipelines. Model #3 ensured the highest maximal test set accuracy but, at the same time, had the highest standard deviation of its performance. We certainly see the differences in classifier performances for different datasets.

However, to study this issue more deeply, we would have to perform a thorough meta-learning analysis which is our plan for the future. Such a study would have to address features of time series in a rigorous manner (for example, with feature extraction techniques proposed by the authors of algorithm catch22: CAnonical Time-series CHaracteristics by Lubba et al. [33] or tsfresh: Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests by Christ et al. [34]).

5.4. Comparisons with state-of-the-art classifiers

In this section, we compare the performance of the proposed approach with the state-of-the-art methods. The selection of the state-of-the-art algorithms was guided by a recent survey by Ruiz et al. [32]. The authors of this survey prepared a comprehensive analysis of the performance of state-of-the-art methods in the multivariate time series classification task. In particular, we compare the proposed approach with random convolutional kernel transform (ROCKET) [35], adaptive dynamic time warping (DTW_A) [36], WEASEL+MUSE [37], Canonical Interval Forest (CIF) [38], multiple representation sequence learner (MrSEQL) [39], HIVE-COTE v1.0 (HC) [17], residual neural network (ResNet) [40], and InceptionTime (IT) [13]. In addition, by analogy to the results given by Ruiz et al. [32], we place results for a "Default" recognition for the majority class.

The absence of some values in Table 6 is intentional. Our comparison utilizes the results given by Ruiz et al. [32]. Ruiz et al. report that it was impossible to compute the results in these cases due to the large size of some datasets.

We compare the best-achieved result with any pair of our Representation and CNN model. The results are collected in Table 6, which presents accuracy and highlights the winning method.

WEASEL+MUSE was overall the most successful classifier — we pose this conclusion based on average accuracy (column-wise averages for Table 6). However, it failed to process six datasets: DuckDuckGeese, EigenWorms, FaceDetection, MotorImagery, PEMS-SF, and PhonemeSpectra, due to high memory complexity. The second best algorithm according to this criterion was ours. WEASEL+MUSE average accuracy of the results collected in Table 6 is 74.54%, while our best approaches achieved an average accuracy of 73.95%. Further ranking of the algorithms according to their average accuracy is Hive Cote, InceptionTime, ROCKET, CIF, MrSEQL, and ResNet. The overall worst performance was achieved by DTW_A.

If we take the number of wins as a performance criterion, we notice that the approach addressed in this paper reports eight wins, which is the best outcome. The second-best were InceptionTime and ROCKET, both with five wins. Then, we have WEASEL+MUSE with four wins.

6. Conclusion

This work is a contribution to the field of time series classification. The essence of our proposal is to transform time series into two-dimensional images and then classify obtained images using a convolutional neural network. Time series with different characteristics were studied in this paper. On average, the highest classification accuracy was achieved using Representation #3, which is based on replicated time series values. It was not an obvious result. Nonetheless, particular decisions were dataset-dependent.

Our methods did not report the best average accuracy when compared to the state-of-the-art methods. We were second-best in this ranking. Nonetheless, the top-performing state-of-the-art classifier, which is WEASEL+MUSE, was not able to finish computations for large datasets. It must be emphasized that we reported the highest count of wins for the considered datasets compared to the state-of-the-art methods. Thus, we may conclude that the proposed approaches are a valid contribution to multivariate time series classification and achieve highly satisfying results.

We have certainly observed substantial differences in performance for various representation/CNN model configurations. It means that we need to search for the best model to obtain good results. From this point of view, our methods may seem a bit less attractive than, say, ROCKET.

Directions for further studies include the following issues. First, a more complete list of transformations, such as, for instance, Val/ChngValChng, is worth attention. Also, three-dimensional images would be tested, e.g., the Val/ValChng/ChngValChng transformation. Second, standardization or normalization of values is often utilized in many problems. These two operations do not bring any new information at all. However, they would affect the results of applied types of neural networks (not only convolutional). Third, more detailed insight into original values would bring valuable results. This hint is based on the observation that transformations using only original values are the best on average. Fourth, polar coordinate systems instead of Cartesian ones could be discussed.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Research was funded by the Warsaw University of Technology, Poland within the Excellence Initiative: Research University (IDUB) programme.

The authors would like to thank Mr. Wiktor Rzemek, Mr. Jakub Stefaniuk and Mr. Przemysław Wcisło. They conducted computational tests in October–December 2021 in frames of a student's project under the supervision of Prof. Władysław Homenda.

[1] Can be downloaded from here: https://www.timeseriesclassification.com/description.php?Dataset=BasicMotions.

[2] Can be downloaded from here: https://www.timeseriesclassification.com/description.php?Dataset=Heartbeat.

Table 6
Accuracy for each dataset. State-of-the-art results are taken from the review paper by Ruiz et al. [32]. Our results concern the best-case scenario (the best-achieved outcome using any pair of representation/CNN model). Def. stands for default result of predicting a majority class.

Dataset	Def.	DTW_A	ROCKET	MUSE	CIF	HC	mrSeql	ResNet	IT	ours
Art.Word.	4.00	98.94	99.56	98.87	97.89	97.99	98.98	98.26	**99.10**	99.00
Atr.Fib.	33.30	22.44	24.89	**74.00**	25.11	29.33	36.89	36.22	22.00	66.66
BasicMot.	25.00	99.92	99.00	**100**	99.75	**100**	94.83	**100**	**100**	**100**
Cricket	8.33	**100**	**100**	99.77	98.38	99.26	99.21	99.40	99.44	94.44
DuckG.	20.00	56.67	46.13		56.00	47.60	39.27	63.20	**63.47**	62.50
Ei.Worms	42.00		86.28		90.33	78.17	72.16	45.45		**97.66**
Epilepsy	26.80	97.37	99.08	99.64	98.38	**100**	99.93	99.18	98.65	96.38
ERing	16.70	92.89	**98.05**	96.89	95.65	94.26	93.19	87.19	92.10	94.82
Eth.Conc.	25.10	29.57	44.68	48.64	72.89	**80.68**	60.18	28.62	27.92	41.83
FaceDet.	50.00	53.13	69.42		68.89	69.17		62.97	**77.24**	67.82
Fing.Mov.	49.00	54.93	55.27	54.77	53.90	53.77	55.53	54.70	56.13	**62.00**
HandMov.	18.90	30.72	44.59	38.02	52.21	37.79	35.23	35.32	42.39	**55.41**
Handwrit.	3.80	60.55	56.67	51.85	35.13	50.41	54.04	59.78	**65.74**	36.47
H.beat	72.20	68.07	71.76	73.59	**76.52**	72.18	72.52	63.89	73.20	70.24
Libras	6.70	87.85	90.61	90.30	91.67	90.28	86.57	**94.11**	88.72	80.56
LSST	31.50	56.96	63.15	**63.62**	56.17	53.84	60.28	42.94	33.97	57.75
Mot.Img.	50.00	50.37	53.13		51.80	52.17	53.00	49.77	51.17	**60.00**
NATOPS	16.70	81.48	88.54	87.13	84.41	82.85	86.43	**97.11**	96.63	96.11
PEMS-SF	11.60	78.73	85.63		**99.85**	97.98	97.15	81.95	82.83	88.44
PenDig.	10.40	99.27	99.56	98.68	98.97	97.19	97.14	99.64	**99.68**	97.54
Phon.Spec.	2.60	15.39	28.35		26.56	32.87		30.86	**36.74**	15.69
Racket.	28.30	85.79	**92.79**	89.56	89.30	90.64	88.73	91.23	91.69	85.53
S.Reg.SCP1	50.20	81.34	86.55	73.58	85.94	86.02	82.86	76.11	84.69	**88.74**
S.Reg.SCP2	50.00	52.43	51.35	49.52	48.87	51.67	49.61	50.24	52.04	**60.00**
St.W.Jump	33.30	25.56	45.56	34.67	45.11	40.67	42.00	30.89	42.00	**60.00**
UWaveG.	12.50	91.51	**94.43**	90.39	92.42	91.31	91.32	88.35	91.23	87.19
avg. acc.	26.88	66.88	72.12	**74.54**	71.94	73.81	71.12	68.20	72.17	73.95
# of wins	0	1	5	4	1	3	3	3	5	**8**

References

[1] W. Homenda, A. Jastrzębska, M. Wrzesień, Time series classification using images, in: R.A. Buchmann, et al. (Eds.), Information Systems Development: Artificial Intelligence for Information Systems Development and Operations, ISD2022 Proceedings, Association for Information Systems, 2022, pp. 1–6.

[2] T. Gorecki, M. Luczak, Using derivatives in time series classification, Data Min. Knowl. Discov. 26 (2013) 310–331, http://dx.doi.org/10.1007/s10618-012-0251-4.

[3] N. Hatami, Y. Gavet, J. Debayle, Bag of recurrence patterns representation for time-series classification, Pattern Anal. Appl. 22 (3) (2019) 877–887, http://dx.doi.org/10.1007/s10044-018-0703-6.

[4] Z. Jia, X. Cai, Z. Jiao, Multi-modal physiological signals based squeeze-and-excitation network with domain adversarial learning for sleep staging, IEEE Sens. J. 22 (4) (2022) 3464–3471, http://dx.doi.org/10.1109/jsen.2022.3140383.

[5] A. Abanda, U. Mori, J.A. Lozano, A review on distance based time series classification, Data Min. Knowl. Discov. 33 (2) (2019) 378–412, http://dx.doi.org/10.1007/s10618-018-0596-4.

[6] B. Bai, G. Li, S. Wang, Z. Wu, W. Yan, Time series classification based on multi-feature dictionary representation and ensemble learning, Expert Syst. Appl. 169 (2021) 114162, http://dx.doi.org/10.1016/j.eswa.2020.114162.

[7] D. Li, T.F. Bissyande, J. Klein, Y.L. Traon, Time series classification with discrete wavelet transformed data, Int. J. Softw. Eng. Knowl. Eng. 26 (09n10) (2016) 1361–1377, http://dx.doi.org/10.1142/S0218194016400088.

[8] M. Arul, A. Kareem, Applications of shapelet transform to time series classification of earthquake, wind and wave data, Eng. Struct. 228 (2021) 111564, http://dx.doi.org/10.1016/j.engstruct.2020.111564.

[9] T. Altay, M.G. Baydogan, A new feature-based time series classification method by using scale-space extrema, Eng. Sci. Technol. Int. J. 24 (6) (2021) 1490–1497, http://dx.doi.org/10.1016/j.jestch.2021.03.017.

[10] P. Schäfer, The BOSS is concerned with time series classification in the presence of noise, Data Min. Knowl. Discov. 29 (6) (2015) 1505–1530, http://dx.doi.org/10.1007/s10618-014-0377-7.

[11] M. Middlehurst, W. Vickers, A. Bagnall, Scalable dictionary classifiers for time series classification, in: Intelligent Data Engineering and Automated Learning, IDEAL 2019, Springer International Publishing, 2019, pp. 11–19, http://dx.doi.org/10.1007/978-3-030-33607-3_2.

[12] 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, CIKM '17, ACM, New York, NY, USA, 2017, pp. 637–646, http://dx.doi.org/10.1145/3132847.3132980.

[13] H.I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D.F. Schmidt, J. Weber, G.I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, InceptionTime: Finding AlexNet for time series classification, Data Min. Knowl. Discov. 34 (2020) 1936–1962, http://dx.doi.org/10.1007/s10618-020-00710-y.

[14] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks, IJCNN, 2017, http://dx.doi.org/10.1109/ijcnn.2017.7966039.

[15] X. Zhang, Y. Gao, J. Lin, C.-T. Lu, TapNet: Multivariate time series classification with attentional prototypical network, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, no. 04, 2020, pp. 6845–6852, http://dx.doi.org/10.1609/aaai.v34i04.6165.

[16] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inform. Sci. 239 (2013) 142–153, http://dx.doi.org/10.1016/j.ins.2013.02.030.

[17] J. Lines, S. Taylor, A. Bagnall, HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, 2016, pp. 1041–1046, http://dx.doi.org/10.1109/ICDM.2016.0133.

[18] M. Flynn, J. Large, T. Bagnall, The contract random interval spectral ensemble (C-RISE): The effect of contracting a classifier on accuracy, Lecture Notes in Comput. Sci. (2019) 381–392, http://dx.doi.org/10.1007/978-3-030-29859-3_33.

[19] A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau, S. Thrun, Dermatologist-level classification of skin cancer with deep neural networks, Nature 542 (7639) (2017) 115–118, http://dx.doi.org/10.1038/nature21056.

[20] S.S. Yadav, S.M. Jadhav, Deep convolutional neural network based medical image classification for disease diagnosis, J. Big Data 6 (1) (2019) http://dx.doi.org/10.1186/s40537-019-0276-2.

[21] S. Lawrence, C. Giles, A.C. Tsoi, A. Back, Face recognition: A convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997) 98–113, http://dx.doi.org/10.1109/72.554195.

[22] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, http://dx.doi.org/10.3115/v1/d14-1181.

[23] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers), 2014, http://dx.doi.org/10.3115/v1/p14-1062.

[24] S. Liu, H. Ji, M.C. Wang, Nonpooling convolutional neural network forecasting for seasonal time series with trends, IEEE Trans. Neural Netw. Learn. Syst. 31 (8) (2020) 2879–2888, http://dx.doi.org/10.1109/tnnls.2019.2934110.

[25] R. Mittelman, Time-series modeling with undecimated fully convolutional neural networks, 2015, http://dx.doi.org/10.48550/ARXIV.1508.00317, arXiv, https://arxiv.org/abs/1508.00317.

[26] B. Zhao, H. Lu, S. Chen, J. Liu, D. Wu, Convolutional neural networks for time series classification, J. Syst. Eng. Electron. 28 (1) (2017) 162–169, http://dx.doi.org/10.21629/jsee.2017.01.18.

[27] D.H. Hubel, T.N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, J. Physiol. 195 (1) (1968) 215–243, http://dx.doi.org/10.1113/jphysiol.1968.sp008455.

[28] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybernet. 36 (4) (1980) 193–202, http://dx.doi.org/10.1007/bf00344251.

[29] K. Yamaguchi, K. Sakamoto, T. Akabane, Y. Fujimoto, A neural network for speaker-independent isolated word recognition, in: ICSLP, 1990.

[30] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551, http://dx.doi.org/10.1162/neco.1989.1.4.541.

[31] B. Williams, M. Toussaint, A. Storkey, Extracting motion primitives from natural handwriting data, in: Artificial Neural Networks – ICANN 2006, in: LNCS, Springer, 2006, http://dx.doi.org/10.1007/11840930_66.

[32] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 35 (2021) 401–449, http://dx.doi.org/10.1007/s10618-020-00727-3.

[33] C.H. Lubba, S.S. Sethi, P. Knaute, S.R. Schultz, B.D. Fulcher, N.S. Jones, catch22: Canonical time-series characteristics, Data Min. Knowl. Discov. 33 (2019) 1821–1852, http://dx.doi.org/10.1007/s10618-019-00647-x.

[34] M. Christ, N. Braun, J. Neuffer, A.W. Kempa-Liehr, Time series feature extraction on basis of scalable hypothesis tests (tsfresh – A Python package), Neurocomputing 307 (2018) 72–77, http://dx.doi.org/10.1016/j.neucom.2018.03.067, https://www.sciencedirect.com/science/article/pii/S0925231218304843.

[35] A. Dempster, F. Petitjean, G.I. Webb, ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels, Data Min. Knowl. Discov. 34 (2020) 1454–1495, http://dx.doi.org/10.1007/s10618-020-00701-z.

[36] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, E. Keogh, Generalizing DTW to the multi-dimensional case requires an adaptive approach, Data Min. Knowl. Discov. 31 (2017) 1–31, http://dx.doi.org/10.1007/s10618-016-0455-0.

[37] P. Schafer, U. Leser, Multivariate time series classification with WEASEL + MUSE, in: 3rdd ECML/PKDD Workshop on AALTD, 2018, https://www2.informatik.hu-berlin.de/~schaefpa/muse.pdf.

[38] M. Middlehurst, J. Large, A. Bagnall, The Canonical Interval Forest (CIF) classifier for time series classification, in: 2020 IEEE International Conference on Big Data, Big Data, 2020, pp. 188–195, http://dx.doi.org/10.1109/BigData50022.2020.9378424.

[39] T.L. Nguyen, S. Gsponer, I. Ilie, M. O'Reilly, G. Ifrim, Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations, Data Min. Knowl. Discov. 33 (2019) 1183–1222, http://dx.doi.org/10.1007/s10618-019-00633-3.

[40] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: A review, Data Min. Knowl. Discov. (2019) http://dx.doi.org/10.1007/s10618-019-00619-1.

Władysław Homenda received the M.Sc. and the Ph.D. degrees from the Warsaw University of Technology, and the D.Sc. degree and the title of Professor from the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. He has been with the Warsaw University of Technology since graduation, where he is currently a Professor with the Faculty of Mathematics and Information Science.

He undertook the organization of graduate studies in computer science with the Centro de Investigacion, Ciencifica y Education, Ensenada, Mexico, the Faculty of Mathematics and Information Science, the Warsaw University of Technolgy, and the Faculty of Mathematics and Informatics, the University of Białystok.

His main research interests are in theoretical foundations of computer science, knowledge representation and processing, and intelligent computing technologies, specifically in the areas of man–machine communication and human–centric computing, fuzzy modeling and granular computing, knowledge discovery, and data mining. He has authored five monographs, more than 130 research articles, and music processing technologies.

Prof. Homenda is currently a member of several editorial boards of international journals.

Agnieszka Jastrzębska received the B.Sc. degree in information technology from the University of Derby, Derby, U.K., in 2009, the M.Sc.Eng. degree in computer engineering from the Rzeszow University of Technology, Rzeszow, Poland, in 2010, the M.A. degree in economics from the University of Rzeszow, Rzeszow, in 2011, and the Ph.D. and D.Sci. degrees from the Warsaw University of Technology, Warsaw, Poland, in 2016 and 2021, respectively. She is an Associate Professor with the Faculty of Mathematics and Information Science, Warsaw University of Technology. Her research interests include machine learning, computational intelligence, and fuzzy modeling.

Prof. Jastrzębska was a recipient of prestigious scholarships from the Institute of Computer Science of Polish Academy of Sciences, the Systems Research Institute of Polish Academy of Sciences, and the Center for Advanced Studies of Warsaw University of Technology. Her Ph.D. dissertation received the Distinguished Dissertation Award. She is Associate Editor of the Applied Soft Computing (Elsevier).

Witold Pedrycz (IEEE Life Fellow) is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada, also with the Systems Research Institute, Polish Academy of Sciences, 00-901 Warsaw, Poland, Istinye University, Faculty of Engineering and Natural Sciences, Department of Computer Engineering, Sariyer/Istanbul, Turkiye. (e-mail: wpedrycz@ualberta.ca). Professor Pedrycz is a foreign member of the Polish Academy of Sciences and a Fellow of the Royal Society of Canada.

He is a recipient of several awards including Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Society, IEEE Canada Computer Engineering Medal, a Cajastur Prize for Soft Computing from the European Centre for Soft Computing, a Killam Prize, a Fuzzy Pioneer Award from the IEEE Computational Intelligence Society, and 2019 Meritorious Service Award from the IEEE Systems Man and Cybernetics Society.

His main research directions involve Computational Intelligence, Granular Computing, and Machine Learning, among others.

Professor Pedrycz serves as an Editor-in-Chief of Information Sciences, Editor-in-Chief of WIREs Data Mining and Knowledge Discovery (Wiley), and Co-editor-in-Chief of Int. J. of Granular Computing (Springer) and J. of Data Information and Management (Springer).

Mariusz Wrzesień is doctor of technical sciences in the discipline of Computer Science, diploma in the specialization of Artificial Decision Systems (Gdańsk University of Technology, 2008), Master of Science in Engineering in the field of Computer Science and Automation (Rzeszów University of Technology, 1999).

In July 1999, he started working as an assistant at the Department of Computer Science, then at the Department of Expert Systems and Artificial Intelligence, and currently at the Department of Artificial Intelligence at the University of Information Technology and Management in Rzeszów. Since the beginning of 2000, he has been conducting research under the scientific supervision of prof. Ph.D. Zdzisław S. Hippe. He is the author/co-author of several published, original scientific papers, presented at national and international scientific conferences. Since February 2009, he became deputy dean of the Computer Science Department, from 2013 to 2019 dean of the Faculty of Applied Information Technology, currently dean of the College of Applied Information Technology.