Classification of Time-Series Images Using Deep Convolutional Neural Networks

Nima Hatami, Yann Gavet and Johan Debayle
Ecole Nationale Superieure des Mines de Saint-Etienne, SPIN/LGF CNRS UMR 5307, 158 cours Fauriel, 42023 Saint-Etienne, France

ABSTRACT
Convolutional Neural Networks (CNN) has achieved a great success in image recognition task by automatically learning a hierarchical feature representation from raw data. While the majority of Time-Series Classification (TSC) literature is focused on 1D signals, this paper uses Recurrence Plots (RP) to transform time-series into 2D texture images and then take advantage of the deep CNN classifier. Image representation of time-series introduces different feature types that are not available for 1D signals, and therefore TSC can be treated as texture image recognition task. CNN model also allows learning different levels of representations together with a classifier, jointly and automatically. Therefore, using RP and CNN in a unified framework is expected to boost the recognition rate of TSC. Experimental results on the UCR time-series classification archive demonstrate competitive accuracy of the proposed approach, compared not only to the existing deep architectures, but also to the state-of-the art TSC algorithms.

Keywords: Convolutional Neural Networks (CNN), Time-Series Classification (TSC), Deep Learning, Recurrence Plots (RP)

1. INTRODUCTION
A time-series is a sequence of data points (measurements) which has a natural temporal ordering. Many important real-world pattern recognition tasks deal with time-series analysis. Biomedical signals (e.g. EEG and ECG), financial data (e.g. stock market and currency exchange rates), industrial devices (e.g. gas sensors and laser excitation), biometrics (e.g. voice, signature and gesture), video processing, music mining, forecasting and weather are examples of application domains with time-series nature.\textsuperscript{4, 5} The time-series analysis motivations and tasks are mainly divided into curve fitting, function approximation, prediction and forecasting, segmentation, classification and clustering categories. In a univariate time-series classification, \( x^n \rightarrow y^n \) so that \( n \)-th series of length \( l \): \( x^n = (x^n_1, x^n_2, ..., x^n_l) \) is associated with a class label \( y^n \in \{1, 2, ..., c\} \). It is worth noting that although this paper is mainly focused on time-series classification problem, the proposed method can be easily adapted to the other tasks such as clustering and anomaly detection.

The existing Time-Series Classification (TSC) methods may be categorized from different perspectives. Regarding the feature types, “frequency-domain” methods include spectral analysis and wavelet analysis; while “time-domain” methods include auto-correlation, auto-regression and cross-correlation analysis. Regarding the classification strategy, it can also be divided into “instance-based” and “feature-based” methods. The former measures similarity between any incoming test sample and the training set; and assigns a label to the most similar class (the euclidean distance based 1-Nearest Neighbor (1-NN) and Dynamic Time Wrapping (DTW) are two popular and widely used methods of this category).\textsuperscript{6, 7} The latter first transforms the time-series into the new space and extract more discriminative and representative features in order to be used by a pattern classifier, which aims of the optimum classification boundaries.\textsuperscript{8, 9}

Recently, Deep Learning (DL also known as feature learning or representation learning) models have achieved a high recognition rate for computer vision\textsuperscript{10, 11} and speech recognition\textsuperscript{12, 13} The Convolutional Neural Networks

Further author information:
N. Hatami: E-mail: nima.hatami@emse.fr, Telephone: +33 477420220
Y. Gavet: E-mail: gavet@emse.fr, Telephone: +33 477420170
J. Debayle: E-mail: debayle@emse.fr, Telephone: +33 477420219
Figure 1: From time-series signal to recurrence plot. Left: A simple example of time-series signal \( (x) \) with 12 data points. Middle: The 2D phase space trajectory is constructed from \( x \) by the time delay embedding (\( \tau = 1 \)). States in the phase space are shown with bold dots: \( s_1 : (x_1, x_2), s_2 : (x_2, x_3), \ldots, s_{11} : (x_{11}, x_{12}) \). Right: The recurrence plot \( R \) is a 11 \( \times \) 11 square matrix with \( R_{i,j} = \text{dist}(s_i, s_j) \).

(CNN) is one of the most popular DL models. Unlike the traditional "feature-based" classification framework, CNN does not require hand-crafted features. Both feature learning and classification parts are unified in one model and are learned jointly. Therefore, their performances are mutually enhanced. Multiple layers of different processing units (e.g. convolution, pooling, sigmoid/hyperbolic tangent squashing, rectifier and normalization) are responsible to learn (represent) a hierarchy of features from low-level to high-level.

This paper investigates the performance of Recurrence Plots (RP) within the deep CNN model for TSC. RP provides a way to visualize the periodic nature of a trajectory through a phase space and enables us to investigate certain aspects of the m-dimensional phase space trajectory through a 2D representation. Because of the recent outstanding results by CNN on image recognition, we first encode time-series signals as 2D plots, and then treat TSC problem as texture recognition task. A CNN model with 2 hidden layers followed by a fully connected layer is used.

2. RELATED WORK

This section briefly reviews the recent deep learning contributions on the TSC task. Application of deep learning on TSC has not been fully explored until recently. There are two main types of approaches, when it comes to application of CNN on TSC: some modify the traditional CNN architecture and use 1D time-series signals as an input, while some others first transform 1D signals into 2D matrices and then apply CNN, similar to the traditional CNN for image recognition. A time delay neural network (TDNN) model is adopted for EEG classification.\(^{15}\) However, their model has only one single hidden layer and is not deep enough to learn hierarchical features. Lee et al.\(^{16}\) explored convolutional deep belief network (CDBN) for audio classification. They applied this unsupervised feature learning on frequency domain rather than in time domain. A multi-channel CNN has been proposed to deal with multivariate time-series.\(^{17}\) Each time series signal is fed into a separate CNN, and afterwards outputs of all CNNs are concatenated and fed into a fully connected MLP classifier. Instead of using the raw signal and learn features automatically, Dalto\(^{18}\) feed CNN with variables post-processed using input variable selection (IVS) algorithm. In\(^{19}\) audio signals are transformed to time-frequency domain and fed into CNN. In another similar work by the same authors, CNN is applied to speech recognition within the framework of hybrid NN-HMM model.\(^{20}\) A multi-scale CNN is proposed for TSC in\(^{21}\). In order to extract features at different scales and frequencies, it transforms signals by down-sampling and smoothing operators, followed by a local convolution stage. A deep CNN is applied on multichannel time-series signals of human activities.\(^{22}\) A sliding window strategy is adopted to put time-series segments into a collection of short pieces of signals. This way, a 2D representation of a 1D time-series signal is obtained and a CNN model applied on 2D matrices (treating them similar to images).

Perhaps\(^{23, 24}\) is the most similar research to our work here. The Gramian Angular Fields (GAF) and Markov Transition Fields (MTF) are used to encode time-series signals as images. Afterwards, a Tiled CNN model is applied to classify the time-series images. Another similar work applied visual descriptors such as Gabor
and Local Binary Patterns (LBP) on RP to extract texture features from time series, and then used SVM classifiers.\textsuperscript{24, 26} The difference between the latter and our work is that they used a traditional classification framework with hand-crafted features and separated feature extraction and classification steps. While our proposed CNN-based framework, automatically learns the texture features that are useful for the classification layer. The joint learning of feature representation and classifier offered by CNN increases the classification performance.

3. METHODOLOGY

3.1 Time-Series to Image Encoding

Time-series can be characterized by a distinct recurrent behavior such as periodicities and irregular cyclicities. Additionally, the recurrence of states is a typical phenomenon for dynamic nonlinear systems or stochastic processes that time-series are generated in. The \textit{R}\textsuperscript{t}\textsuperscript{14} is a visualization tool that aims to explore the m-dimensional phase space trajectory through a 2D representation of its recurrences. The main idea is to reveal in which points some trajectories return to a previous state and it can be formulated as:

\[
R_{i,j} = \theta(\epsilon - \| \vec{s}_i - \vec{s}_j \|), \quad \vec{s}(.) \in \mathbb{R}^m, \quad i, j = 1, ..., K
\]  

where \( K \) is the number of considered states \( \vec{s}_i \), \( \epsilon \) is a threshold distance, \( \| . \| \) a norm and \( \theta(\cdot) \) the Heaviside function. The R-matrix contains both texture which are single dots, diagonal lines as well as vertical and horizontal lines; and typology information which are characterized as homogeneous, periodic, drift and disrupted. For instance, fading to the upper left and lower right corners means that the process contains a trend or drift; or vertical and horizontal lines/clusters show that some states do not change or change slowly for some time and this can be interpreted as laminar states.\textsuperscript{27} Obviously, there are patterns and information in RP that are not always very easy to visually see and interpret.

Figure 1 shows the step-by-step instructions for calculating RP for a simple time-series signal. First the 2D phase space trajectory (\( m = 2 \)) is constructed from the time-series. Then, the R-matrix is calculated based on the closeness of the states in the phase space. It is worth noting that the resulting R-matrix by Formula 1 has only \{0,1\} values, that caused by thresholding parameter \( \epsilon \). In order to avoid information loss by binarization of the R-matrix, this paper skips the thresholding step and uses the gray-level texture images. Inspired by the unique texture images obtained from the R-matrices, this paper proposed a TSC pipeline based on the CNN model. First the raw 1D time-series signals \( x^n \) are transformed into 2D recurrence texture images, and then both features and classifier are jointly learned in one unified model.

3.2 CNN for Time-Series Image Classification

There are two aspects of any CNN model that should be considered carefully: i) designing an appropriate architecture, and ii) choosing the right learning algorithm. Both architecture and learning rules should be
Figure 3: Application of RP \( m = 3, \tau = 4 \) time-series to image encoding on five different datasets from the UCR archive: 50words, TwoPatterns, FaceAll, OliveOil and Yoga data (from left to right, respectively).

chosen in a way that they are not only compatible with each other, but also fit the data and the application appropriately.

**Architecture.** A 2-stage deep CNN model is applied here with 1-channel input of size 28 × 28 and the output layer with \( c \) neurons. Each feature learning stage is representing a different feature-level and consists of convolution (filter), activation, and pooling operators, respectively. The input and output of each layer are called feature maps. A filter layer convolves its input with a set of trainable kernels. The convolutional layer is the core building block of a CNN and exploits spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. The connections are local, but always extend along the entire depth of the input volume in order to produce the strongest response to a spatially local input pattern. The activation function (such as sigmoid and tanh) introduces non-linearity into the networks and allows them to learn complex models. Here we applied ReLU (Rectified Linear Units) because it trains the neural networks several times faster without a significant penalty to generalisation accuracy. Pooling (a.k.a. subsampling) reduces the resolution of input and make it robust to small variations for previously learned features. It combines the outputs of \( i-1 \) th layer into a single input in \( i \) th layer over a range of local neighborhood.

At the end of the 2-stage feature extraction, the feature maps are flatten and fed into a fully connected (FC) layer for classification. FC layers connect every neuron in one layer to every neuron in another layer, which in principle are the same as the traditional multi-layer perceptron (MLP). The proposed pipeline for TSC is shown in Figure 2.

**Learning.** Training the above CNN architecture is similar to the MLPs. Gradient-based optimization method (error back-propagation algorithm) is utilized to estimate parameters of the model. For faster convergence, the stochastic gradient descent (SGD) is used for updating the parameters. The training phase has two main steps: propagation and weight update. Each propagation involves feedforward and error back-propagation passes. Former determines the feature maps on input vector by passing from layer to layer until reaching the output (left to right in Figure 2). Latter, calculates the propagation errors according to the loss function for the predicted output (error propagates from right to left in Figure 2). Predicted error on each layer is used for calculating the derivatives by taking advantage of chain-rule of derivative. Once the derivatives of parameters obtained, the weight is updated as follows: the weight’s output delta and input activation are multiplied to find the gradient of the weight. And then, a ratio (learning rate) of the weight’s gradient is subtracted from the weight. This learning cycle is repeated until the network reaches a satisfactory validation error. More details on CNN architecture and learning algorithm can be found in.

**4. EXPERIMENTS**

In order to evaluate the performance of the proposed method, the UCR time-series classification archive is used. For having a comprehensive evaluation of the classification algorithms, this repository contains 85 time-
Table 1: Performance (in terms of the error rates) of the proposed method compared to the state-of-the-art TSC algorithms on 20 selected data from the UCR archive.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>c</th>
<th>Nte</th>
<th>Ntr</th>
<th>l</th>
<th>1-NN DTW</th>
<th>Shapelet</th>
<th>BoP</th>
<th>SAX-VSM</th>
<th>TFRP</th>
<th>MCNN</th>
<th>GAF-MTF</th>
<th>ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>50words</td>
<td>50</td>
<td>455</td>
<td>270</td>
<td>0.31</td>
<td>0.44</td>
<td>0.46</td>
<td>-</td>
<td>0.38</td>
<td>0.20</td>
<td>0.23</td>
<td>0.37</td>
<td>0.28</td>
</tr>
<tr>
<td>Acient</td>
<td>37</td>
<td>391</td>
<td>176</td>
<td>0.39</td>
<td>0.51</td>
<td>0.43</td>
<td>0.38</td>
<td>0.20</td>
<td>0.23</td>
<td>0.37</td>
<td>0.28</td>
<td>0.28</td>
</tr>
<tr>
<td>Beef</td>
<td>5</td>
<td>30</td>
<td>470</td>
<td>0.36</td>
<td>0.44</td>
<td>0.43</td>
<td>0.033</td>
<td>0.36</td>
<td>0.36</td>
<td>0.23</td>
<td>0.28</td>
<td>0.08</td>
</tr>
<tr>
<td>CBF</td>
<td>3</td>
<td>30</td>
<td>900</td>
<td>0.003</td>
<td>0.05</td>
<td>0.01</td>
<td>0.02</td>
<td>-</td>
<td>0.002</td>
<td>0.009</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>Coffee</td>
<td>2</td>
<td>28</td>
<td>286</td>
<td>0</td>
<td>0.06</td>
<td>0.03</td>
<td>0</td>
<td>0.03</td>
<td>0.036</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ECG200</td>
<td>2</td>
<td>100</td>
<td>96</td>
<td>0.23</td>
<td>0.22</td>
<td>0.14</td>
<td>0.14</td>
<td>0.17</td>
<td>-</td>
<td>0.09</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FaceAll</td>
<td>14</td>
<td>560</td>
<td>1690</td>
<td>0.19</td>
<td>0.40</td>
<td>0.21</td>
<td>0.20</td>
<td>0.29</td>
<td>0.23</td>
<td>0.23</td>
<td>0.19</td>
<td>0</td>
</tr>
<tr>
<td>Face4</td>
<td>4</td>
<td>24</td>
<td>88</td>
<td>0.17</td>
<td>0.09</td>
<td>0.023</td>
<td>0</td>
<td>0.21</td>
<td>0</td>
<td>0.06</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fish</td>
<td>7</td>
<td>175</td>
<td>463</td>
<td>0.17</td>
<td>0.19</td>
<td>0.074</td>
<td>0.017</td>
<td>0.12</td>
<td>0.05</td>
<td>0.114</td>
<td>0.085</td>
<td>0</td>
</tr>
<tr>
<td>GnutPoint</td>
<td>2</td>
<td>50</td>
<td>150</td>
<td>0.093</td>
<td>0.061</td>
<td>0.027</td>
<td>0.007</td>
<td>0.02</td>
<td>0</td>
<td>0.08</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lighting2</td>
<td>2</td>
<td>60</td>
<td>61</td>
<td>0.13</td>
<td>0.29</td>
<td>0.16</td>
<td>0.19</td>
<td>0.04</td>
<td>0.16</td>
<td>0.11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lighting7</td>
<td>7</td>
<td>70</td>
<td>73</td>
<td>0.27</td>
<td>0.40</td>
<td>0.46</td>
<td>0.30</td>
<td>0.31</td>
<td>0.21</td>
<td>0.26</td>
<td>0.26</td>
<td>0</td>
</tr>
<tr>
<td>OliveOil</td>
<td>4</td>
<td>30</td>
<td>30</td>
<td>0.16</td>
<td>0.21</td>
<td>0.13</td>
<td>0.10</td>
<td>0.13</td>
<td>0.13</td>
<td>0.2</td>
<td>0.11</td>
<td>0</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>6</td>
<td>200</td>
<td>242</td>
<td>0.40</td>
<td>0.35</td>
<td>0.23</td>
<td>0.107</td>
<td>0.07</td>
<td>0.27</td>
<td>0.35</td>
<td>0.29</td>
<td>0</td>
</tr>
<tr>
<td>SwedishLeaf</td>
<td>15</td>
<td>500</td>
<td>625</td>
<td>0.20</td>
<td>0.27</td>
<td>0.19</td>
<td>0.25</td>
<td>0.04</td>
<td>0.066</td>
<td>0.06</td>
<td>0.06</td>
<td>0</td>
</tr>
<tr>
<td>SyntControl</td>
<td>6</td>
<td>300</td>
<td>300</td>
<td>0.007</td>
<td>0.08</td>
<td>0.03</td>
<td>0.25</td>
<td>-</td>
<td>0.003</td>
<td>0.007</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trace</td>
<td>4</td>
<td>100</td>
<td>275</td>
<td>0</td>
<td>0.002</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TwoPattern</td>
<td>4</td>
<td>1000</td>
<td>128</td>
<td>0</td>
<td>0.11</td>
<td>0.12</td>
<td>0.004</td>
<td>-</td>
<td>0.002</td>
<td>0.09</td>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>Wafer</td>
<td>2</td>
<td>1000</td>
<td>152</td>
<td>0.02</td>
<td>0.004</td>
<td>0.003</td>
<td>0.0006</td>
<td>0.002</td>
<td>0.002</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Yoga</td>
<td>2</td>
<td>300</td>
<td>426</td>
<td>0.16</td>
<td>0.24</td>
<td>0.17</td>
<td>0.16</td>
<td>0.14</td>
<td>0.11</td>
<td>0.19</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># wins</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Ave.Rank</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.05</td>
<td>5.75</td>
<td>4.15</td>
<td>3.0</td>
<td>3.37</td>
<td>2.36</td>
<td>3.40</td>
<td>2.15</td>
<td>2.15</td>
</tr>
</tbody>
</table>

The code for the proposed model is available on: [http://sites.google.com/site/nimahatami/projects](http://sites.google.com/site/nimahatami/projects)
the kernel size, $S_1$ and $S_2$ are pooling size, $H$ and $O$ represent the number of hidden and output neurons in MLP. Based on this template, the proposed CNN model denoted as $32(5)-2-32(5)-2-128$-c. The loss function of "categorical-crossentropy" with "adam" optimizer is used. The batch and epoch sizes are chosen from {5, 20} and {50, 250, 1000, 2000}, respectively and based on their performance on the validation set. Since finding the optimal CNN parameters is still an open problem, we followed the rules of thumb to choose the parameters. Visualization of kernels in the 1st and 2nd representation layers from a CNN trained for "TwoPattern" data is given in Figure 4. Error rate of the proposed model is given in Table 1. Besides comparing with $1–NNDTW$, three state-of-the-art TSC algorithms are also reported: Fast-Shapelets, Bag-of-Patterns (BoP also known as BoF or BoW), and SAX-VSM. Not all the performances are reported on the selected data. Therefore, there are some missing elements (shown with ")") in the Table. We have also compared our results with two recent deep CNN models, i.e. Multi-scale CNN (MCNN) and GAF-MTF. The former performs various transformations of the time-series which extract features of different frequency and time scales embedded in a CNN framework. The latter, which has the similar pipeline to ours, first encodes time-series to images using GAF and MTF, and then applies a Tiled CNN. As shown, the proposed model obtains 10 out of 20 wins and average rank of 2.15. The second most accurate algorithm is MCNN (6 wins out of 19, and Ave.Rank = 2.36), followed by SAX-VSM (6 wins out of 19, and Ave.Rank = 3.0). The other deep CNN model that uses time-series images (GAF-MTF) ranks 5th. It highlights the fact that choosing a right time-series encoding is crucial for CNN model. Furthermore, it is observable that TFRP algorithm using RP images in a traditional classification framework with different texture descriptors (followed by a SVM classifier) ranks better than GAF-MTF. It can be explained that the RP introduces more discriminant and informative features than the GAF-MTF transform for TSC. Additionally, comparing the proposed method with TFRP highlights the fact that for TSC of RP images the CNN model is more effective than the traditional classification frameworks.

5. CONCLUSIONS

A novel pipeline for TSC is proposed. Taking advantage of the CNN’s high performance on image classification, time-series signals are first transformed into texture images (using RP) and then handled by a deep CNN model. This pipeline offers the following advantages: i) RP enables us to visualize certain aspects of the m-dimensional phase space trajectory through a 2D images, and ii) CNN automatically learns different levels of time-series features and classification jointly and in a supervised manner. Experimental results demonstrate the superiority of the proposed pipeline. In particular, comparing with models using RP with the traditional classification framework (e.g. SIFT, Gabor and LBP features with SVM classifier) and other CNN-based time-series image classification (e.g. GAF-MTF images with CNN) demonstrates that using RP images with CNN in our proposed model obtains the better results.

As future work, CNN architecture with more feature representation layers should be investigated for more difficult TSC tasks (preferably with more data samples available). Large datasets are needed in order to train a deeper architectures. Therefore, adopting the proposed pipeline for TSC with small sample sizes can be another interesting future direction. Exploring different ensemble learning methods for CNN can be also interesting. We will particularly be investigating application of the output coding for CNNs.
ACKNOWLEDGMENTS

This research is partially supported by the French national research agency (ANR) under the PANDORE grant with reference number ANR-14-CE28-0027.

REFERENCES


