Journal of Systems Engineering and Electronics
Vol. 28, No. 1, February 2017, pp.162 – 169

Convolutional neural networks for time series classification

Bendong Zhao*, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu

College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China

Abstract: Time series classification is an important task in time series data mining, and has attracted great interests and tremendous efforts during last decades. However, it remains a challenging problem due to the nature of time series data: high dimensionality, large in data size and updating continuously. The deep learning techniques are explored to improve the performance of traditional feature-based approaches. Specifically, a novel convolutional neural network (CNN) framework is proposed for time series classification. Different from other feature-based classification approaches, CNN can discover and extract the suitable internal structure to generate deep features of the input time series automatically by using convolution and pooling operations. Two groups of experiments are conducted on simulated data sets and eight groups of experiments are conducted on real-world data sets from different application domains. The final experimental results show that the proposed method outperforms state-of-the-art methods for time series classification in terms of the classification accuracy and noise tolerance.

Keywords: time series, multivariate time series, classification, convolutional neural network (CNN), data mining.

DOI: 10.21629/JSEE.2017.01.18

1. Introduction

Time series is an important class of temporal data objects, and can be easily obtained by recording a series of observations chronologically, e.g. price of stocks, electrocardiogram (ECG), and brightness of a target star [1]. Recently, time series data mining has attracted great interests and initiated various researches [2 – 6]. The main tasks of time series data mining include, but are not limited to: pattern discovery and clustering, rule discovery, summarization and classification [1]. As an important task of time series data mining, time series classification is widely used in many areas. For instance, in astronomy, the brightness of a target star can be classified with a set of references in order to search for exoplanets [7]. In medical science, the blood pressure or ECG classification can be used to help diagnosing cardiac disorders [8,9]. And in computer science, time series classification can be used for signature verification and speech recognition [10]. There are many approaches for time series classification, which can be summarized in three large categories according to the classification scheme: model based, distance based and feature based [7,11].

The first category of time series classification approaches is based on models. Assume that the time series in a class is generated by an underlying model, and then the model parameters are fitted and determined by the training samples in this class. In this way, different models (different model parameters) are obtained for different classes. Then a new time series is compared to the models to determine which class it belongs to. The autoregressive (AR) model is the simplest method and has been commonly used for time series analyses [12]. However, this method has a drawback that the time series must satisfy stationary assumption, which is always violated in practice. The Markov model (MM) and the hidden Markov model (HMM) can be used to model non-stationarity time series. They have been frequently used and have obtained good performances in time series classification in recent years [13]. However, there is another serious drawback of these two models that the time series needs to be symbolic. As a matter of fact, most of time series cannot, or is very difficult to, be represented well and truly by a generative model, which highly limits the application of these model based approaches.

The second category of time series classification approaches is based on distance. After defining a distance function to measure the similarity (or dissimilarity) between two time series, many existing methods, e.g. K nearest neighbor (KNN), support vector machines (SVM), can be directly used for classification. Thus, the key point of these approaches is how to define the distance function. For two time series $\boldsymbol{x}$, $\boldsymbol{y}$, the simplest distance function may be L_p norm, which is defined as

$$L_p(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_p = (\sum_i |x_i - y_i|^p)^{1/p}. \quad (1)$$

Manuscript received December 09, 2015.
*Corresponding author.

when $p = 1$, $p = 2$, $p = \infty$, it denotes Manhattan distance, Euclidean distance and maximum distance, respectively. L_p norm has been widely used to measure the dissimilarity between two time series due to its simplicity. However, there is a serious drawback of this method that the length of the two series must be equal and it is sensitive to distortions in time dimension [11]. To overcome this drawback, another distance function called dynamic time warping (DTW) distance is proposed. DTW has always been a research hotspot in time series data mining and many achievements can be found in [4,14 – 16]. However, the DTW distance does not satisfy the triangle inequality, and is computationally expensive.

The third category of time series classification approaches, also most widely used, is based on features. The main idea is to achieve dimension reduction by using a set of features representing the time series. The earliest spectral method of discrete Fourier transform (DFT) transforms a time series from time domain to frequency domain, and then uses a few Fourier coefficients to represent the original data. It can maintain most information of the time series as well as highly reduce its dimension, because the energy of a signal is always concentrated on low-frequencies [7]. However, DFT is a global method and it will lose many local features of the original data. To overcome the disadvantage, windowed Fourier transform (WFT), also called short-time Fourier transform (STFT), is proposed, while its drawback is that it is difficult to determine the window size. And then discrete wavelet transform (DWT) is proposed. DWT is a kind of time-frequency multi-resolution analysis method and is widely used for time series analysis. However, it can only suit the time series with length of 2^k. Beside the spectral method, another category of the dimension reduction method is based on eigenvalue. The representative eigenvalue analysis methods include principal component analysis (PCA), singular value decomposition (SVD) and sparse coding [17]. Recently, the shapelet has been studied as a new feature for time series data mining and obtained good performance in classification tasks [18 – 21]. As a matter of fact, the performance of all the feature based methods highly depends on the feature designing.

In recent years, deep learning has gained incredible popularity and many achievements can be found in literature [17,22 – 25]. As one of deep networks, convolutional neural network (CNN) has been successfully used in object recognition. German traffic sign recognition benchmark (GTSRB) is a challenge match in 2011 International Joint Conference on Neural Networks, with an accepted database included. And in this match, CNN obtained the best performance. Instead of human-designed features, CNN can automatically mine and generate deep features of input images, and has a strong robustness against translation, scaling and rotation, owing to the three important ideas different from traditional forward neural networks as follows: local receptive field, weights sharing and pooling [26].

Inspired by the CNN for image recognition, we explore a deep learning framework for time series classification. Specifically, a novel CNN framework for time series classification is proposed. Convolution and pooling operations are alternatively used to generate deep features of the raw data. Then the features are connected to a multilayer perceptron (MLP) to perform classification. Experimental results on simulated and real data sets show that our method outperforms state-of-the-art methods for time series classification in terms of the classification accuracy and noise tolerance. The remaining part of this paper is organized as follows. In Section 2, the architecture and training algorithm of CNN for time series classification are described. The meanings and the determination methods of several CNN parameters are discussed in Section 3. Experiments and conclusions are shown in Sections 4 and 5 respectively.

2. The methodology

In this section, the basic CNN architecture and its training algorithm for time series classification are described.

2.1 CNN architecture for time series classification

CNN is usually composed of two parts. In part 1, convolution and pooling operations are alternatively used to generate deep features of the raw data. And in part 2, the features are connected to an MLP for classification. In the previous work of CNN for multivariate time series classification, Zheng et al. proposed a multi-channels deep convolution neural network [27]. They separated multivariate time series into univariate ones for feature learning individually and obtained good classification performance in experiments. However, there is a major limitation that this method cannot mine the interrelationship between different univariate time series. To overcome this defect, we modify the algorithm in this way: instead of feature learning individually, the multivariate time series is jointly trained for feature extraction. To be better understood, a typical CNN architecture for three-variate time series classification with two convolutional layers and two pooling layers is illustrated in Fig. 1.

Here are some details for each layer:

(i) Input layer. Input layer has $N \times k$ neurons, where k denotes the variate number of input time series and N denotes the length of each univariate series.

(ii) Convolutional layer. Perform convolution operations on the time series of preceding layer with convolution filters. Here are some filter parameters to be determined previously according to domain knowledge or just depending on experiments, such as, filter numbers m, convolution stride s and the size of filter $k \times l$, where k denotes the variate number of the time series in the preceding layer and l denotes the length of filter. A nonlinear transformation function f also needs to be determined in this layer. For instance, if the preceding layer contains k-variate time series and the length of each univariate is N, after the convolutional operation, we get m-variate time series and the length of each univariate is $\left\lfloor \dfrac{N-l}{s}+1 \right\rfloor$, where $\lfloor \cdot \rfloor$ denotes rounding down.

(iii) Pooling layer. A feature map is divided into N equal-length segments, and then every segment is represented by its average or maximum value. The advantage of pooling operation is down-sampling the convolutional output bands, thus reducing variability in the hidden activations [26].

(iv) Feature layer. After several convolution and pooling operations, the original time series is represented by a series of feature maps. Simply connect all the feature maps to generate a new long time series as the final representation of the original input in the feature layer.

(v) Output layer. The output layer has n neurons, corresponding to n classes of time series. It is fully connected to the feature layer. The most popular method is taking the maximum output neuron as the class label of the input time series in classification task.

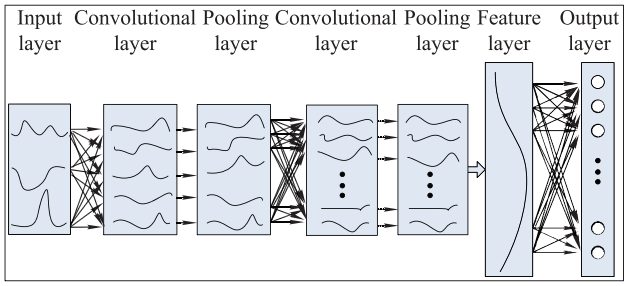


Fig. 1 CNN architecture for three-variate time series classification

2.2 Training of CNN

The CNN is trained via a sequence of training examples $((\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_{Nsample}, \boldsymbol{y}_{Nsample}))$ with $\boldsymbol{x}_t \in \mathbf{R}^{N\times k}$, $\boldsymbol{y}_t \in \mathbf{R}^n$ for $1 \leqslant t \leqslant Nsample$. The multivariate or univariate time series $\boldsymbol{x}_t$ is given as input to the network, while the vector $\boldsymbol{y}_t$ denotes the target output. The network is trained according to the following several steps:

Step 1 Initialize the network. Determine the CNN architecture, composed of two convolutional layers and two pooling layers, as shown in Fig. 1. Fix the neuron number of input layer and output layer according to the classification task. Set all the CNN parameters, as described in Section 2.1. Initialize the weights and bias with a small random number. Select a learning rate η, and an activation function f, and the commonly used example is the sigmoid function:

$$f(x) = sigmoid(x) = \frac{1}{1+e^{-x}}. \tag{2}$$

Step 2 Choose a training sample from the training set randomly.

Step 3 Calculate the output of each layer.

(i) The output of the convolutional layer can be written as

$$\boldsymbol{C}_r(t) = f(\sum_{i=1}^{l}\sum_{j=1}^{k} \boldsymbol{x}(i+s(t-1), j)\boldsymbol{\omega}_r(i,j) + \boldsymbol{b}(r)) \tag{3}$$

where $\boldsymbol{x} \in \mathbf{R}^{N\times k}$ denotes the input time series or the output of the preceding layer, s denotes the convolution stride, $\boldsymbol{C}_r(t)$ refers to the tth component of the rth feature map, $\boldsymbol{\omega}_r \in \mathbf{R}^{l\times k}$ and $\boldsymbol{b}(r)$ refer to the weights and bias of the rth convolution filter.

(ii) The output of the pooling layer can be written as

$$\boldsymbol{P}_r(t) = g(\boldsymbol{C}_r((t-1)l+1), \boldsymbol{C}_r((t-1)l+2), \ldots, \boldsymbol{C}_r(tl)) \tag{4}$$

where the function g represents the pooling strategy, the most popular used is averaging or max pooling. It is obvious that pooling operation achieves reducing the point data, while not changing the number of feature maps.

(iii) The output of the output layer can be written as

$$\boldsymbol{O}(j) = f(\sum_{i=1}^{M} \boldsymbol{z}(i)\boldsymbol{\omega}_f(i,j) + \boldsymbol{b}_f(j)), \quad j = 1, 2, \ldots, n \tag{5}$$

where $\boldsymbol{z}$ denotes the final feature map in the feature layer, $\boldsymbol{b}_f$ is the bias of the output layer and $\boldsymbol{\omega}_f \in \mathbf{R}^{M\times n}$ refers to the connection weights between the feature layer and the output layer.

So the mean-square error can be written as

$$E = \frac{1}{2}\sum_{k=1}^{n} \boldsymbol{e}(k)^2 = \frac{1}{2}\sum_{k=1}^{n} (\boldsymbol{O}(k) - \boldsymbol{y}(k))^2. \tag{6}$$

Step 4 Update the weights and bias by the gradient descent method.

$$\boldsymbol{p} = \boldsymbol{p} - \eta\frac{\partial E}{\partial \boldsymbol{p}} \tag{7}$$

where $\boldsymbol{p}$ is the value of the parameter, and $\boldsymbol{p}$ refers to $\boldsymbol{\omega}_r$, $\boldsymbol{\omega}_f$, $\boldsymbol{b}$, or $\boldsymbol{b}_f$ in this CNN.

Step 5 Choose another training sample and go to Step 3 until all the samples in the training set have been trained.

Step 6 Increase the iteration number. If the iteration number is equal to the maximum value which is set previously, terminate the algorithm. Otherwise, go to Step 2.

3. Analysis of CNN parameters

In this section, the meaning of three important parameters is discussed, including the convolutional filter size, the pooling method and the number of convolution filters. And a set of experiments are designed to help determining an appropriate CNN architecture for classification on the simulated univariate time series which will be described in Section 4.1.

3.1 Convolutional filter size

The convolutional filters actually represent the local features of input time series or feature maps. If the filter size is too small, it cannot well represent the typical features of waveforms. However it will be difficult to reflect local features conversely. Table 1 shows the classification results for different filter sizes. The signal to noise ratio (SNR) of test data is set as 5 dB, which is defined as (8). It is suggested that seven is the most reasonable filter size for both of two convolutional layers.

$$\mathrm{SNR} = 10 \lg \frac{P_S}{P_N} \tag{8}$$

where P_S represents the signal power, and P_N is the variance of Gaussian white noise.

Table 1 Classification results for different filter sizes

Filter size of the first convolutional layer	Filter size of the second convolutional layer	Error/%
5	5	5.75
5	7	2.25
7	5	3.75
7	7	0.75
7	9	1.25
9	7	1.75

3.2 Pooling method

The purpose of pooling operation is to achieve dimension reduction of feature maps, while preserving most information. The pooling size is an important parameter to be determined previously. The lager the pooling size is, the better performance it obtains in dimension reduction, however the more information it loses. Besides, the pooling strategy also needs to be designed before training, the commonly used are max-pooling and mean-pooling. Table 2 shows the classification results for different pooling sizes and pooling strategies. It is suggested that three is the most reasonable pooling size with the mean-pooling strategy.

Table 2 Classification results for different pooling methods

Pooling size	Pooling strategy	Error/%
2	Mean-pooling	0.75
2	Max-pooling	0.75
3	Mean-pooling	0.5
3	Max-pooling	0.63
4	Mean-pooling	0.75
5	Mean-pooling	1.5

3.3 Number of convolution filters

The filters actually represent the local features of time series according to aforementioned analysis. Obviously too few filters cannot extract enough features to achieve classification tasks. However, more filters are helpless when the filters are already enough to represent the discriminative features to achieve classification. Moreover, more filters will make the training of CNN more computationally expensive. Table 3 shows the classification results for different filter numbers of the two convolutional layers. The results indicate that with the increasing of the filter number, the classification performance gets better at first, and then maintains steady, however the training time keeps rapid growing all the time. A reasonable choice of the filter number is 6 for the first convolutional layer and 12 for the second.

Table 3 Classification results for different filter number

Filter number of the first convolutional layer	Filter number of the second convolutional layer	Training time/s	Error/%
2	4	196	8.5
3	6	326	6.75
6	9	742	1.25
6	12	1096	0.5
6	15	1328	0.5
9	12	1592	1.75

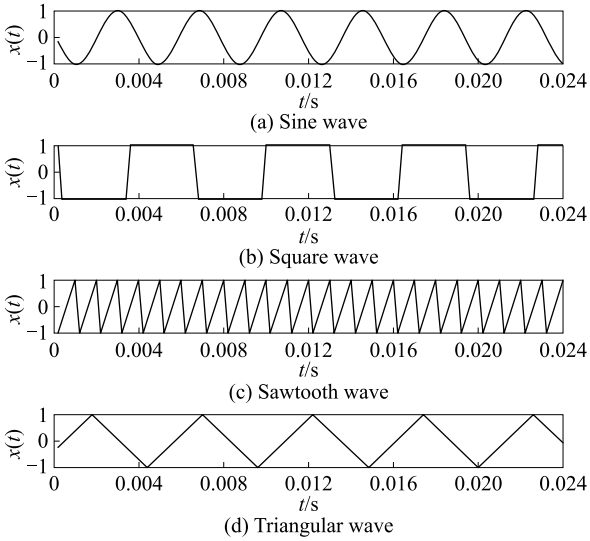
4. Experiments

In this section, to prove the advantage of our proposed method, we conduct several groups of experiments on simulated univariate, simulated multivariate and real-world time series data sets respectively. The experimental data sets are firstly described, then the evaluation and comparison methods are introduced, the results and discussion are presented finally.

4.1 Experimental data

4.1.1 Simulated univariate time series

The following four kinds of waveforms are chosen as the objects for classification: sine wave, square wave, sawtooth wave and triangular wave, as shown in Fig. 2.



(a) Sine wave
(b) Square wave
(c) Sawtooth wave
(d) Triangular wave

Fig. 2 Examples of simulated univariate time series without noise

The training set and test set are obtained respectively as follows.

(i) Training set. The frequency of the waveforms is restricted from 120 Hz to 1 100 Hz, stepped by 20 Hz and sampled at 5 kHz. The initial phase of each waveform is restricted from 0.5π to 2π, stepped by 0.5π and the maximum amplitude is restricted to 1. In this way, altogether 800 waveforms are obtained as the training set.

(ii) Test set. Generate 200 waveforms for each class, frequency and initial phase of each waveform are randomly chosen from the interval [100 Hz, 1 kHz] and $[0, 2\pi)$ respectively. Totally 800 waveforms are obtained as the test set.

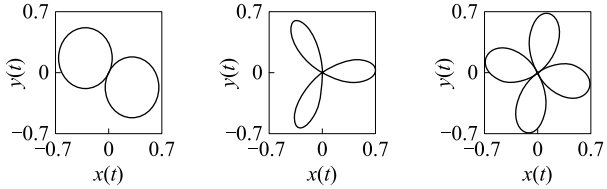
4.1.2 Simulated multivariate time series

We choose the same simulated data set as Hüsken et al. described in [28], as shown in Fig. 3. The simulated multivariate time series used in our experiments stem from sampling two-dimensional trajectories of three different classes, as (9)–(11) described. α is chosen as 0.7 to bound the input time series between -0.7 and 0.7, and β is randomly drawn for every time series from the interval $[0, 2\pi)$. To receive the time series, each trajectory is sampled with a step size of $2\pi/120$, starting from a randomly chosen value $t_0 \in [0, 2\pi)$. Generate 500 trajectories for each class and randomly choose 300 of them for training and the rest 200 for test.

$$\text{Class 1}: \begin{cases} x(t) = \alpha \sin(t+\beta)|\sin(t)| \\ y(t) = \alpha \cos(t+\beta)|\sin(t)| \end{cases} \tag{9}$$

$$\text{Class 2}: \begin{cases} x(t) = \alpha \sin(\dfrac{1}{2}t+\beta)\ \sin(\dfrac{3}{2}t) \\ y(t) = \alpha \cos(t+\beta)\sin(2t) \end{cases} \tag{10}$$

$$\text{Class 3}: \begin{cases} x(t) = \alpha \sin(t+\beta)\sin(2t) \\ y(t) = \alpha \cos(t+\beta)\sin(2t) \end{cases} \tag{11}$$



(a) Class 1 (b) Class 2 (c) Class 3

Fig. 3 Examples of simulated multivariate time series without noise

4.1.3 Real-word time series

"UCR Time Series Data Mining Archive" is a publicly available time series data set and has always been used for classification. As CNN is a deep learning method which needs large amounts of samples to train the network, eight data sets are chosen as our objects for classification, which are presented in Table 4. A linear transformation is used for data preprocessing to restrict each time series ranging from -1 to 1.

Table 4 Summary of data sets

Data set	Train case	Test case	Length	Class
Swedish Leaf	500	625	128	15
Wafer	1 000	6164	153	2
ECG200	100	100	96	2
Two Patterns	1 000	4 000	128	4
Face All	560	1 690	131	14
Synthetic Control	300	300	60	6
Yoga	300	3 000	426	2
OSU Leaf	200	242	427	6

4.2 Evaluation and comparison methods

The main purpose of experiments on simulated data sets is to prove the algorithm's ability of noise tolerance. Traditional methods such as 1-NN are not suitable for noise-adding data sets, so two learning algorithms called support vector machines (SVM) and deep belief networks (DBN) are employed as our comparison methods.

For the real-world data sets originating from the "UCR Time Series Data Mining Archive", we choose two commonly used methods named 1-NN and SVM as our baseline methods. In recent years, two new methods called multiclass SVM with weighted dynamic time warping kernel function (MSVMWDTWK) [29] and derivative DTW (DDTW) [30] are proposed to improve time series classification performance. And they are also chosen as our comparison methods.

In all of our experiments, the CNN parameters are experimentally determined as aforementioned analysis in Section 3. The architecture of DBN is well set to obtain its best performance. Restricted Boltzmann machine (RBM) is used for pre-training for each layer of DBN. And LIBSVM-3.20 is used in this paper.

4.3 Experimental results

4.3.1 Results on simulated time series

To verify the noise tolerance of CNN, two kinds of measures are taken in the experiments. On the one hand, we use the training set with different noise levels (2 dB, 6 dB and 10 dB) to train different CNNs, DBNs and SVMs. And on the other hand, we use the test set with different noise levels (0 dB to 10 dB, stepped by 0.5 dB) to test the CNN, DBN and SVM trained by the same training set. The classification results on simulated univariate and multivariate time series are shown in Fig. 4 and Fig. 5 respectively.

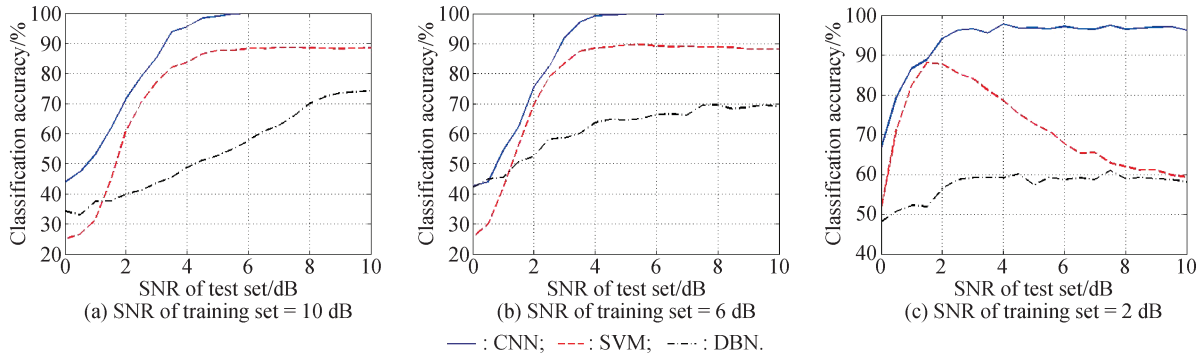


Fig. 4 Classification results on simulated univariate time series

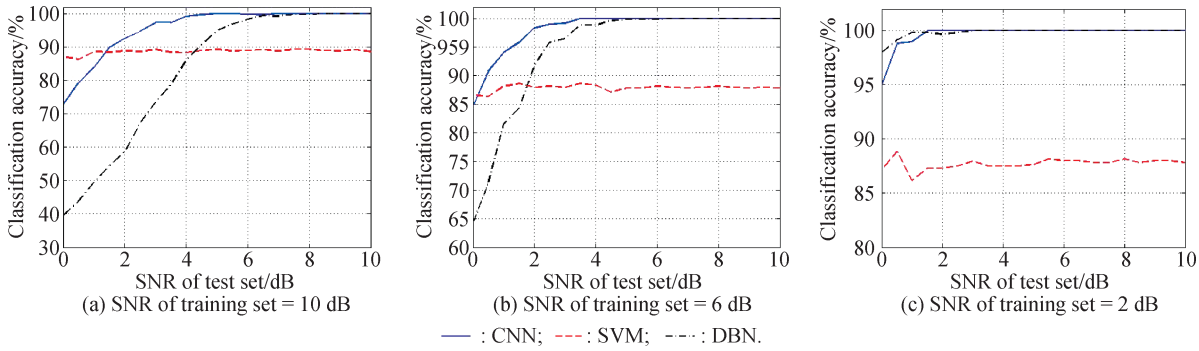


Fig. 5 Classification results on simulated multivariate time

4.3.2 Results on real-world time series

To verify the classification ability of CNN, we conduct eight groups of experiments on real-world data sets originating from the "UCR Time Series Data Mining Archive". And the main results are shown in Table 5. The number in parentheses after classification accuracy denotes the performance rank of the five different methods.

Table 5 Classification results (and rank) on real-world time series

Data set	SVM	1-NN	Jeong's method	Górecki's method	CNN
Swedish Leaf	65.8(5)	82.6(3)	75.4(4)	88.2(2)	88.7(1)
Wafer	98.6(3)	99.6(2)	89.0(5)	98.0(4)	99.7(1)
ECG200	79.0(5)	88.0(2)	84.0(3)	83.0(4)	89.0(1)
Two Patterns	84.5(5)	89.7(4)	98.3(2)	99.7(1)	96.5(3)
Face All	58.7(5)	67.9(4)	74.4(3)	91.2(2)	91.5(1)
Synthetic Control	97.0(3)	77.0(5)	91.2(4)	99.3(1)	97.3(2)
Yoga	57.1(5)	83.3(3)	82.0(4)	85.6(2)	86.2(1)
OSU Leaf	41.3(5)	54.1(4)	87.5(2)	87.6(1)	81.2(3)

4.4 Discussion

As shown in Fig. 4 and Fig. 5, the experimental results indicate that no matter for simulated univariate or multivariate time series and no matter how much SNR of training set is, CNN obtains the best classification performance comparing to SVM and DBN. The main reason is that CNN can discover and extract the suitable internal structure to generate deep features of the raw data automatically by using convolution and pooling operations. Moreover, the deep features are robust against translation and scaling. For the four classes of simulated univariate time series, the variation of frequency and initial phases can be seen as scaling and translation of the time series respectively. And for the three classes of simulated multivariate time series, the variation of starting time and initial phase can be seen as arbitrary translation of the curves. Therefore, CNN can obtain the best classification performance beyond all doubt. Besides, the results also demonstrate that CNN obtains the best anti-noise performance.

As shown in Table 5, in the eight groups of experiments on real-world data sets originating from the "UCR Time Series Data Mining Archive", CNN obtains the best clas-

sification performances on most datasets. Although the experimental performances of CNN are not in the first rank on three groups of the data sets, which are "Two Patterns", "Synthetic Control" and "OSU Leaf", their classification accuracies are close to those first-ranking algorithms. After all, it demonstrates that CNN is the most competitive candidate for time series classification.

5. Conclusions

In this paper, we propose a novel CNN-based method for time series classification. Instead of using human-designed features, CNN can automatically discover and extract the internal structure of the input time series to generate deep features for classification. We evaluate our proposed method on two simulated data sets and eight real-world data sets. Experimental results show that the CNN method outperforms the competing baseline methods in terms of classification accuracy and noise tolerance. It also demonstrates that the deep learning method can learn more robust deep features, which is helpful to improve time series classification performance. Besides, the meaning of three CNN parameters is discussed in this paper, including the convolutional filter size, the pooling method and the number of convolution filters. Nevertheless, there are some limitations to this work: (i) the training of CNN is time consuming as most of parameters are determined by lots of experiments and (ii) the length of time series must be fixed during training and testing which is determined by the architecture of CNN. To solve these problems, the method of designing optimal parameters and the study on a new network architecture combining CNN with recursive neural network (RNN) are ongoing.

References

[1] T. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 2011, 24(1): 164 – 181.

[2] R. Al-Hmouz, W. Pedrycz, A. Balamash, et al. Description and classification of granular time series. *Soft Computing*, 2015, 19(4): 1003 – 1017.

[3] Z. Chen, W. Zuo, Q. Hu, et al. Kernel sparse representation for time series classification. *Information Sciences*, 2015, 292(1): 15 – 26.

[4] H. Li. On-line and dynamic time warping for time series data mining. *International Journal of Machine Learning and Cybernetics*, 2015, 6(1): 145 – 153.

[5] I. López-Yáñez, L. Sheremetov, C. Yáñez-Márquez. A novel associative model for time series data mining. *Pattern Recognition Letters*, 2014, 41(1): 23 – 33.

[6] H. Li. Asynchronism-based principal component analysis for time series data mining. *Expert Systems with Applications*, 2014, 41(6): 2842 – 2850.

[7] H. Kaya, Ş. Gündüz-öğüdücü. A distance based time series classification framework. *Information Systems*, 2015, 51(C): 27 – 42.

[8] J. Wang, P. Liu, M. F. H. She, et al. Bag-of-words representation for biomedical time series classification. *Biomedical Signal Processing and Control*, 2013, 8(6): 634 – 644.

[9] Y. Özbay, R. Ceylan, B. Karlik. A fuzzy clustering neural network architecture for classification of ECG arrhythmias. *Computers in Biology and Medicine*, 2006, 36(4): 376 – 388.

[10] O. Abdel-Hamid, A. Mohamed, H. Jiang, et al. Convolutional neural networks for speech recognition. *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, 2014, 22(10): 1533 – 1545.

[11] Z. Xing, J. Pei, E. Keogh. A brief survey on sequence classification. *Special Interest Group Knowledge Discovery and Data Mining Explorations Newsletter*, 2010, 12(1): 40 – 48.

[12] B. V. Kini, C. C. Sekhar. Large margin mixture of AR models for time series classification. *Applied Soft Computing*, 2013, 13(1): 361 – 371.

[13] A. Antonucci, R. De Rosa, A. Giusti, et al. Robust classification of multivariate time series by imprecise hidden Markov models. *International Journal of Approximate Reasoning*, 2015, 56(B): 249 – 263.

[14] R. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 2016, 30(2): 1 – 30.

[15] T. Rakthanmanon, B. Campana, A. Mueen, et al. Addressing big data time series: mining trillions of time series subsequences under dynamic time warping. *ACM Trans. on Knowledge Discovery from Data*, 2013, 7(3): 1 – 31.

[16] Y. Jeong, M. K. Jeong, O. A. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 2011, 44(9): 2231 – 2240.

[17] Y. Bengio, A. Courville, P. Vincent. Representation learning: a review and new perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2013, 35(8): 1798 – 1828.

[18] J. Zakaria, A. Mueen, E. Keogh, et al. Accelerating the discovery of unsupervised-shapelets. *Data Mining and Knowledge Discovery*, 2016, 30(1): 243 – 281.

[19] M. Arathi, A. Govardhan. Effect of mahalanobis distance on time series classification using shapelets. *Proc. of the 49th Annual Convention of the Computer Society of India*, 2015: 525 – 534.

[20] J. Hills, J. Lines, E. Baranauskas, et al. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 2014, 28(4): 851 – 881.

[21] L. Ye, E. Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 2011, 22(1/2): 149 – 182.

[22] M. Längkvist, L. Karlsson, A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 2014, 42(1): 11 – 24.

[23] Y. Bengio. Deep learning of representations: looking forward. *Proc. of the International Conference on Statistical Language and Speech*, 2013: 1 – 37.

[24] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends ®in Machine Learning*, 2009, 2(1): 1 – 127.

[25] J. Schmidhuber. Deep learning in neural networks: an overview. *Neural Networks*, 2015, 61(1): 85 – 117.

[26] P. Swietojanski, A. Ghoshal, S. Renals. Convolutional neural networks for distant speech recognition. *IEEE Signal Processing Letters*, 2014, 21(9): 1120 – 1124.

[27] Y. Zheng, Q. Liu, E. Chen, et al. Time series classification using multi-channels deep convolutional neural networks. *Proc. of the 15th International Conference on Web-Age Information Management*, 2014: 298 – 310.

[28] M. Hüsken, P. Stagge. Recurrent neural networks for time series classification. *Neurocomputing*, 2003, 50(1): 223 – 235.

[29] Y. Jeong, R. Jayaraman. Support vector-based algorithms with

weighted dynamic time warping kernel function for time series classification. *Knowledge-Based Systems*, 2015, 75(C): 184 – 191.

[30] T. Górecki, M. łuczak. Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, 2013, 26(2): 310 – 331.

Biographies

Bendong Zhao was born in 1991. He received his B.S. degree in science and technology of electronic information from National University of Defense Technology (NUDT), Changsha, China, in 2013. He is currently a Ph.D. candidate at the Laboratory of Automatic Target Recognition, NUDT. His research interests include time series data mining, target recognition and machine learning.
E-mail: zhaobendong_nudt@163.com

Huanzhang Lu was born in 1963. He received his M.S. and Ph.D. degrees from National University of Defense Technology (NUDT), Changsha, China, in 1990 and 1994 respectively. He is currently a full professor at the Laboratory of Automatic Target Recognition, NUDT. His research interests include automatic target recognition, precision and real time system.
E-mail: luhz@nudt.edu.cn

Shangfeng Chen was born in 1978. He received his M.S. and Ph.D. degrees from National University of Defense Technology (NUDT), Changsha, China, in 2004 and 2009 respectively. He is currently a lecturer at the Laboratory of Automatic Target Recognition, NUDT. His research interests include automatic target recognition and real time system.
E-mail: cactus510@sina.com

Junliang Liu was born in 1988. He received his M.S. degree in science and technology of electronic information from National University of Defense Technology (NUDT), Changsha, China, in 2013. He is currently a Ph.D. candidate at the Laboratory of Automatic Target Recognition, NUDT. His research interests include optical guidance, target recognition and signal processing.
E-mail: liujunliang_1988@163.com

Dongya Wu was born in 1990. She received her M.S. degree in science and technology of electronic information from National University of Defense Technology (NUDT), Changsha, China, in 2014. She is currently a Ph.D. candidate at the Laboratory of Automatic Target Recognition, NUDT. Her research interests include optical guidance, target recognition and signal processing.
E-mail: wudongya1226@163.com