Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 9549323, 7 pages
https://doi.org/10.1155/2017/9549323

Research Article

A Cycle Deep Belief Network Model for Multivariate Time Series Classification

Shuqin Wang,[1,2] Gang Hua,[1] Guosheng Hao,[2] and Chunli Xie[2]

[1]*School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221116, China*
[2]*School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, Jiangsu 221116, China*

Correspondence should be addressed to Gang Hua; ghua@cumt.edu.cn

Received 27 May 2017; Revised 11 July 2017; Accepted 24 August 2017; Published 4 October 2017

Academic Editor: M. L. R. Varela

Multivariate time series (MTS) data is an important class of temporal data objects and it can be easily obtained. However, the MTS classification is a very difficult process because of the complexity of the data type. In this paper, we proposed a Cycle Deep Belief Network model to classify MTS and compared its performance with DBN and KNN. This model utilizes the presentation learning ability of DBN and the correlation between the time series data. The experimental results showed that this model outperforms other four algorithms: DBN, KNN_ED, KNN_DTW, and RNN.

1. Introduction

Time series data are sequences of real-valued signals that are measured at successive time intervals. They can be divided into two kinds: univariate time series and multivariate time series (MTS). Univariate time series contain one variable, while MTS have two or more variables. MTS is a more important data type of time series because it is widely used in many areas such as speech recognition, medicine and biology measurement, financial and market data analysis, telecommunication and telemetry, sensor networking, motion tracking, and meteorology.

As the availability of MTS data increases, the problem of MTS classification attracts great interest recently in the literature [1]. MTS classification is a supervised learning procedure aimed for labeling a new multivariate series instance according to the classification function learned from the training set [2]. However, the features in traditional classification problems are independent of their relative positions, while the features in time series are highly correlated. That resulted in the loss of some important information if the traditional classification algorithms are used for MTS, since they treat each feature as an independent attribute. Many techniques have been proposed for time series classification. A method based on boosting are presented for multivariate time series classification [3]. In [4], the authors proposed a DTW based decision tree to classify time series and the error rate is 4.9%. In [5], the authors utilize a multilayer perceptron neural network on the control chart problem and the best performance achieved is 1.9% error rate. Hidden Markov Models are used on the PCV-ECG classification problem and achieve 98% accuracy [6]. Support vector machine combined with Gaussian Elastic Metric Kernel is used for time series classification [7]. The dynamics of recurrent neural networks (RNNs) for the classification of time series are presented in [8]. However, simple combination of one-nearest-neighbor with DTW distance is claimed to be exceptionally difficult to beat [9].

Deep Belief Network is a type of deep neural network with multiple hidden layers, introduced by Hinton et al. [10] along with a greedy layer-wise learning algorithm. Restricted Boltzmann Machine (RBM), a probabilistic model, is the building block of DBN. DBN and RBM have witnessed increased attention from researchers. They have already been applied in many problems and gained excellent performance, such as classification [11], dimensionality-reduction [12], and information retrieval [13]. Taylor et al. [14] proposed conditional RBM, an extension of the RBM, which is applied to human notion sequences. Chao et al. [15] evaluated the DBN performance as a forecasting tool on predicting exchange rate. Längkvist et al. [16] applied DBN for sleep stage classification and evaluated the performance. The result illustrated that DBN either with features (feat-DBN) or using the raw data (raw-DBN) performed better than the feat-GOHMM. The feat-DBN achieved 72.2% and the raw-DBN achieved 67.4%, while the feat-GOHMM achieved only 63.9%.

Raw-DBN do not need to extract feature before classifying the sleep data and this algorithm is easy to implement. However, it neglects the important information in time series data and its performance is not satisfactory. This paper proposed a Cycle DBN model for time series classification. This model possesses the ability of feature learning since it is developed on the basis of DBN. Meanwhile, the characters of time series data are taken into consideration in the model.

The remainder of the paper is organized as follows. Next section reviews the background material. In Section 3, we detail the Cycle DBN model for multivariate time series. Section 4 evaluates the performance of our Cycle DBN on two real data sets. Section 5 concludes the work of this paper.

2. Background Material

A time series is a sequence of observations over a period of time. Formally, a univariate time series $x = \{x(i) \in R : i = 1, 2, \dots, n\}$ is an ordered set of n real-valued numbers, and n is called the length of the time series x. Multivariate time series is more common in real life and it is more complex since it has two or more variables. A MTS is defined as a finite sequence of univariate time series

$$X = (x_1, x_2, \dots, x_m). \quad (1)$$

The MTS X has m variables and the corresponding component of the jth variable x_j is a univariate time series of length n:

$$x = \left\{x_j(i) \in R : i = 1, 2, \dots, n\right\} \quad (j = 1, 2, \dots, m). \quad (2)$$

In this paper, we use bold face characters for MTS and regular fonts for univariate time series.

The time series classification problem is a supervised learning procedure. First we should learn a function $f : X \to y$ according to the given training set $A = \{(X^{(i)}, y^{(i)})\}$ $i = 1, 2, \dots, k$. The training set A includes k samples and each sample consists of an input $X^{(i)}$ paired with its corresponding label $y^{(i)}$. Then we can assign a label to a new time series instance based on the function we learned from the training set.

A Deep Belief Network (DBN) consists of an input layer, a number of hidden layers, and finally an output layer. The top two layers have undirected, symmetric connections between them. The lower layers receive top-down, directed connections from the layer above.

The process of training DBNs includes two phases. Each two consecutive layers in DBN are treated as a Restricted Boltzmann Machine with visible units v and hidden units h. There are full connections between visible layer and hidden layer, but no visible-to-visible or hidden-to-hidden connections (see Figure 1). The visible and hidden units are connected with a weight matrix, W, and have a visible bias vector b and a hidden bias vector c, respectively. We need to train each RBM independently one after another and then stack

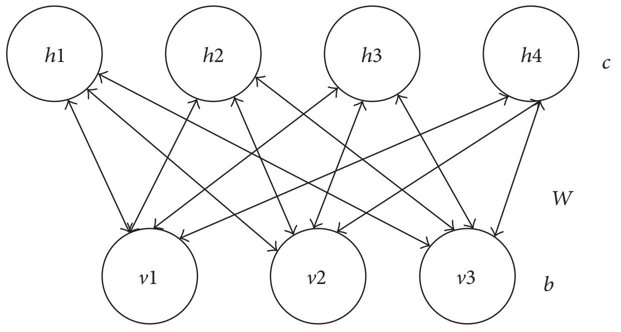

FIGURE 1: Graphical depiction of RBM.

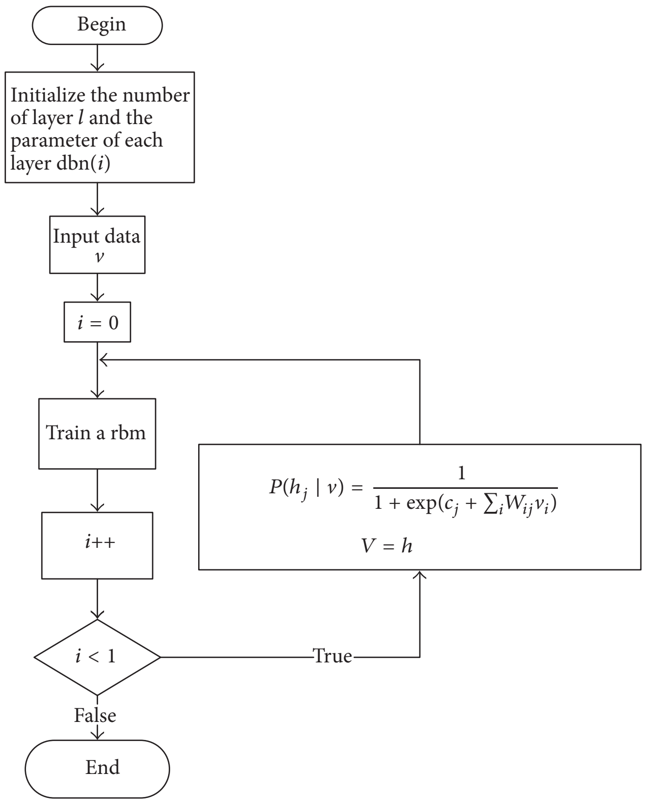
Begin
Initialize the number of layer l and the parameter of each layer dbn(i)
Input data v
$i = 0$
Train a rbm
$P(h_j \mid v) = \frac{1}{1 + \exp(c_j + \sum_i W_{ij} v_i)}$
$V = h$
i++
$i < 1$
True
False
End

FIGURE 2: The flowchart of DBN training.

them on top of each other in the first phase. This procedure is also called pretraining. In the second phase, the BP network is set up at the last level of the DBN, and the output of the highest RBM is received as its input. Then we can perform a supervised learning in this phase. This procedure is called fine-tuning since the parameters in the DBN are tuned using error back propagation algorithm in this phase.

The procedure of training DBN is shown by Algorithm 1 and the corresponding flowchart is given by Figure 2.

From the above analysis, we can conclude that the most important of DBN is the training of each RBM.

Since there are no hidden-hidden or visible-visible connections in the RBM, the probability that hidden unit h_j is activated by visible vector $P(h_j|v)$ and the probability that

```
Begin
Initialize $\alpha$ (the learning rate), $l$ (the number of
DBN layers), $m_k$ (the number of hidden unites in $k$ layer
and $k = 1, 2, \ldots, l$), epochs, DBN($k$).parameters
(the parameters of each layer and $k = 1, 2, \ldots, l$).
    Input data $X$;
    $V1 = X$;
    $k = 1$;
    do
    trainRBM($V1$, DBN($k$).parameters);
    $k$++;
    compute $h$ according to Equation (3)
    $V1 = h$;
    while ($k < l$)
    End.
```

Algorithm 1: DBN train algorithm.

visible unit v_i is activated by given hidden vector $P(v_i|h)$ is given by

$$P\left(h_j \mid v\right) = \frac{1}{1 + \exp\left(c_j + \sum_i w_{ij} v_i\right)}, \tag{3}$$

$$P\left(v_i \mid h\right) = \frac{1}{1 + \exp\left(b_i + \sum_j w_{ji} h_j\right)}. \tag{4}$$

Contrastive Divergence (CD) approximation is used to train the parameters by minimizing the reconstruction error and the learning rule is given by

$$\frac{\partial \log P(v)}{\partial W_{ij}} \approx \left\langle v_i h_j \right\rangle_{\text{data}} - \left\langle v_i h_j \right\rangle_{\text{recon}}. \tag{5}$$

$\langle v_i h_j \rangle_{\text{data}}$ is expectation of the training set and $\langle v_i h_j \rangle_{\text{recon}}$ represents the expectation of the distribution of reconstructions.

The procedure of training RBM is shown as Algorithm 2 and the corresponding flowchart is given by Figure 3.

```
Begin
m = 1;
while (m < epoch)
for all hidden units $j$
$P\left(h1_j \mid v1\right) = \frac{1}{1 + \exp\left(c_j + \sum_i W_{ij} v1_i\right)}$
Sample $h1_i \in \{0, 1\}$ from $P(h1_j|v1)$
End for
For all visible units $i$ do
$P\left(v2_i \mid h1_j\right) = \frac{1}{1 + \exp\left(b_i + \sum_j W_{ij} h1_j\right)}$
Sample $v2_i \in \{0, 1\}$ from $P(v2_j \mid h1)$
End for
For all hidden units $j$ do
$P\left(h2_j \mid v2\right) = \frac{1}{1 + \exp\left(c_j + \sum_i W_{ij} v2_i\right)}$
$h2_j = P\left(h2_j \mid v2\right)$
end for
$w = w + \alpha * (h1 * v1' - h2 * v2')$
$b = b + \alpha * (v1 - v2)$
$c = c + \alpha * (h1 - h2)$
End while
End.
```

Algorithm 2: The algorithm for RBM train.

3. Cycle_DBN for Time Series Classification

Längkvist et al. [16] applied DBN in time series classification and obtained a remarkable result. The standard DBN optimizes the posterior probability $p(y_t \mid x_t)$ of the class labels given the current input xt. However, time series data are different from other kinds of data and there are correlations between time series data. It is unsuitable to apply DBN for time series classification without any modification because it neglects the important information in time series data.

Based on the above discussion, this paper proposed a Cycle DBN model for time series classification just as Figure 4. The model inherits the powerful feature representation of DBN and utilizes the data correlation of the time series. Thus, this model is quite suitable for time series classification.

In this model, X_t is the input at time step t and O_t is the corresponding output of DBN. Since our purpose is classification, we add a softmax function on the top layer and y_t is the corresponding label. After training DBN and getting the label y_t, y_t is then treated as one item input of DBN. At time t, the inputs of DBN not only include X_t but also include y_{t-1}, the output of DBN at time $t-1$.

The training procedure of this Cycle_DBN, which is similar to the traditional DBN, includes two procedures. The only difference is that the output at time $t-1$ is feedback to Cycle_DBN as one of the inputs at time t. The first procedure is unsupervised training to initiate the parameters of DBN. After unsupervised learning, we add a softmax function on the top layer and do a supervised training procedure.

4. Experimental Evaluation

In this section, we conduct extensive experiments to evaluate the classification performance of the proposed model Cycle_DBN and compare it against traditional DBN, KNN_ED, KNN_DTW, and recurrent neural networks (RNN).

The k-NN is one of the most well-known classification algorithms that are very simple to understand but performs well in practice. An object in the testing set is classified according to the distances of the object to the objects in the training set and the object is assigned to the class its k nearest neighbors belongs to. We will choose $k = 1$ in our experiment and the algorithm is simply called the nearest neighbor algorithm. In KNN_ED, we use Euclidean Distance to measure the similarity between two instances.

Dynamic Time Warping (DTW) [17] is another distance measure for time series and it was originally and typically designed for univariate time series. However, the time series handled in this paper is multidimensional and a multidimensional version of DTW is needed. Fortunately, ten

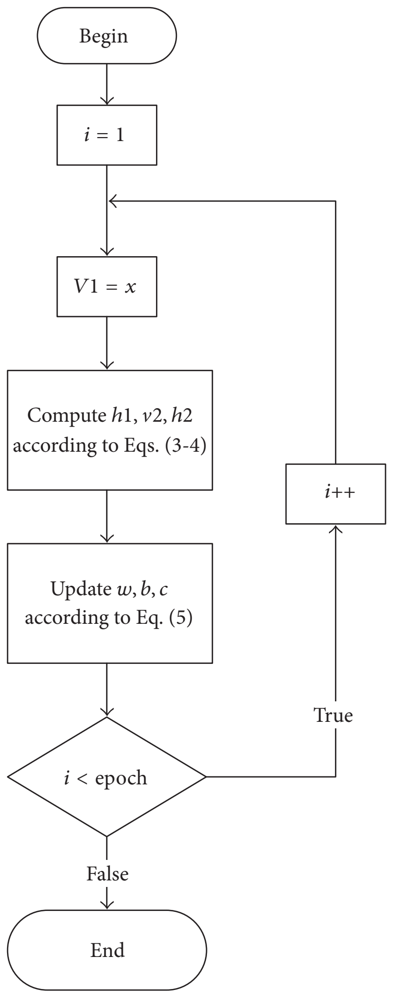
Begin
$i = 1$
$V1 = x$
Compute $h1, v2, h2$ according to Eqs. (3-4)
Update w, b, c according to Eq. (5)
$i < \text{epoch}$
True
$i{+}{+}$
False
End

Figure 3: The flowchart of RBM training.

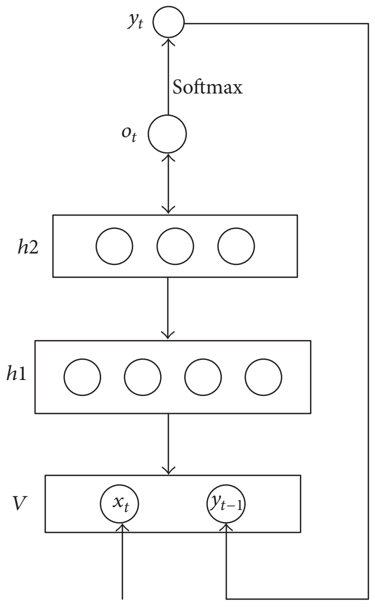

Figure 4: The architecture of Cycle DBN.

Holt et al. [18] proposed a multidimensional DTW and it utilizes all dimensions to find the best synchronization. In standard DTW, the distance is usually calculated by taking the squared distance between the feature values of each combination of points: $d(q_i, c_j) = (q_i - c_j)^2$. But in multidimensional DTW, a distance measure for two K-dimensional points must be calculated: $d(q_i, c_j) = \sum_{k=1}^{K}(q_{ik} - c_{jk})^2$. In KNN_DTW, we use multidimensional DTW distance to measure the similarity between two instances.

RNN allows the identification of dynamic system with an explicit model of time and memory, which makes it ideal for time series classification. In this paper, we choose Elman's architecture, which consist of a context layer, an input layer, one hidden layer, and an output layer.

To evaluate the performance of these methods, we test them on real-world time series datasets, including sleeping dataset, PAMAP2 dataset, and UCR Time Series Classification Archive.

The performance of the classifier is reported using error rate and the error rate of classifiers is defined as shown in

$$\text{error\_rate} = \frac{\text{total number of misclassification data}}{\text{total number of testing data}}. \tag{6}$$

4.1. Sleep Stage Classification. We first consider the problem of sleep stage classification. The data used in the paper is provided by St. Vincent's University Hospital and University College Dublin and can be downloaded from http://www.physionet.org/pn3/ucddb/ PhysioNet.

4.1.1. Dataset. The recordings of this data set have been obtained from 25 adult subjects with suspected sleep-disordered breathing. Each recording consists of 2 EEG channels (C3-A2 and C4-A1), 2 EOG channels, and 1 EMG channel. We only use one of the EEG signals (C3-A2) in our study

$$X_t = (\text{EEG}_t \;\; \text{EOG1}_t \;\; \text{EOG2}_t \;\; \text{EMG}_t). \tag{7}$$

According to Rechtschaffen and Kales (R&K) [19], sleep recordings can be divided into the following five stages: awake, rapid eye movement (REM), stage 1, stage 2, and slow wave sleep (SWS). Our goal is to find a map function f that correctly predicts the corresponding sleep stage according to the $\mathbf{X}_t$: $y_t = f(X_t)$.

4.1.2. Experiment Setup. The raw signals of all subjects are slightly preprocessed by notch filtering at 50 Hz to cancel out power line disturbances and then are prefiltered with a band-pass filter of 0.3 to 32 Hz for EEG and EOG and 10 to 32 Hz for EMG. After that they are downsampled to 64 Hz.

Since the sample rate is 64 samples per second and we set window width w to be 1 second of data, our time series become

$$X_i = \left(\text{EEG}_{1+i}{}^{64+i}, \text{EOG1}_{1+i}{}^{64+i}, \text{EOG2}_{1+i}{}^{64+i}, \text{EMG}_{1+i}{}^{64+i}\right). \tag{8}$$

Table 1: Distribution of five classes in the data set.

Subject	Total	Awake	REM	Stage 1	Stage 2	SWS
trainSamples	25000	5017	5005	4993	4986	4999
Val Samples	5000	983	995	1007	1014	1001
Ucdbb009	20000	4230	3720	6420	740	4890
Ucdbb010	20000	1980	11040	2610	2480	1890
Ucdbb011	20000	3270	6170	2430	390	7740
Ucdbb012	20000	4380	7710	930	3660	3320
Ucdbb013	20000	2670	4040	3660	2010	7620
Ucdbb014	20000	0	6780	7380	1190	4650

Table 2: Classification error rate of five models on sleep datasets.

Subject	Cycle_DBN	DBN	*K*NN_ED	*K*NN_DTW	RNN
Ucdbb009	**0.0061**	0.27619	0.3572	0.3359	0.432568
Ucdbb010	**0.0112**	0.19108	0.45832	0.3971	0.5609
Ucdbb011	**0.0044**	0.10805	0.58243	0.57314	054586
Ucdbb012	**0.0098**	0.04804	0.41244	0.3379	0.45141
Ucdbb013	**0.0068**	0.19709	0.4964	0.4857	0.65820
Ucdbb014	**0.0077**	0.26535	0.47192	0.36183	0.58422

Since the length of $\mathbf{X}_t$ is 64, we have corresponding 64 labels. The last label is selected as the label of the time series $\mathbf{X}_t$.

In our study, we use five people recordings as the training set. In order to balance the samples, we select 6000 records every category random. So we have 30000 recordings and we divide 25000 into train samples and 5000 into validation samples. The other six people recordings are used for test data. The distribution of dataset is listed in Table 1.

4.1.3. Experiment Result. Our goal is to compare the performance of IDBNs with original DBN, *k*NN_ED, *K*NN_DTW, and RNN for time series classification. We illustrate the error rate of each model in Table 2. The best results are recorded in boldface in Table 2.

Compared with other four algorithms, the proposed algorithm has best performance. The classification accuracies of Cycle_DBN on all the test data are up to 90% and especially most of them are more than 99%. Standard DBN has a higher rate of correct classification than *K*NN_ED, *K*NN_DTW, and RNN. RNN shows quite poor performance and the error rate is about 50%.

4.2. Activity Classification. Our second experiment is on the PAMAP2 dataset for activity classification. This dataset can be downloaded at http://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring.

4.2.1. Dataset. This data set records 18 activities performed by 9 subjects wearing 3 IMUs and a HR-monitor. Each of data contains 54 columns per row and the columns contain the following data: timestamp (1), activityID (2), heart rate (3), IMU hand (4–20), IMU chest (21–37), and IMU ankle (38–54). In our experiment, we only select 7 activities which are "lying (1)," "sitting (2)," "standing (3)," "walking (4)," "running (5)," "cycling (6)," "Nordic walking (7)." Since the records of subject103 and subject109 do not have all the above seven activities and we discard these two subjects. That is to say, we select subject 101~subject 102 and subject 104~subject 108, seven subjects to classify seven activities. Furthermore, the record of heart rate is not used in our experiment.

4.2.2. Experiment Setup. To improve the performance of the proposed approach, we need to carry out a data preprocessing process at the beginning of the experiment. Each dimension of time series is normalized through

$$x = \frac{x - \text{mean}(x)}{\text{std}(x)}, \tag{9}$$

where mean(x) and std(x) are the mean and standard deviation of the variable for samples belonging to the same column, not all samples.

For each subject of seven subjects, we randomly select 1/2 as training set, 1/6 as validation set, and the rest as test set.

4.2.3. Experiment Result. We evaluate classification accuracies of each model on these seven subjects. Table 3 shows the detailed error rates comparison of each subject. From Table 3 we can see that the classification accuracies of the five models on the seven datasets are more than 90%. However, our Cycle_DBN model is either the lowest error rate one or very close to the lowest error rate one for each subject. *K*NN_ED also shows quite excellent performance and we should note that *K*NN is feature-based model.

It is well known that feature-based models have an advantage over lazy classification models such as *K*NN in efficiency. Although *K*NN has high classification accuracy, the prediction time of *K*NN will increase dramatically when the size of training data set grows. The prediction time of DBN and Cycle_DBN will not increase no matter how large the training data is. Therefore, Cycle_DBN shows excellent

Table 3: Classification error rate of five models on PAMAP2.

Subject	Cycle_DBN	DBN	KNN_ED	KNN_DTW	RNN
Subject 101	**5.70E − 05**	0.01208	0.000171	0.03	0167081
Subject 102	1.82E − 05	0.000675	**0**	0.034	0.036484
Subject 104	**0**	0.000197	3.93E − 05	0.006	0.000995
Subject 105	3.38E − 05	0.004746	**0**	0.024	0.101161
Subject 106	**1.82E − 05**	0.001546	3.64E − 05	0.018	0
Subject 107	**0**	0.000438	3.99E − 05	0.024	0.062604
Subject 108	3.48E − 05	0.000869	**1.74E − 05**	0.018	0.0833

Table 4: Classification error rate of five models on ten UCR time series datasets.

Dataset	Cycle_DBN	DBN	KNN_ED	KNN_DTW	RNN
uWaveGestureLibrary_Z	**0.333333**	**0.330656**	0.350363	0.35	0.419598
UWaveGestureLibraryAll	0.095047	0.07095	**0.051926**	0.052	0.100503
FordA	0.489647	0.487211	**0.341016**	0.341	0.484865
Two Patterns	0.044365	**0.032374**	0.09325	0.090	0.10775
ECG5000	**0.049161**	0.051559	0.075111	0.075	0.072889
Wafer	**0.000838**	0.002513	0.004543	0.005	0.006976
StarLightCurves	**0.069481**	0.070779	0.151166	0.151	0.326979
ElectricDevices	**0.029931**	0.338983	0.449228	0.376	0.9135
InsectWingbeatSound	0.39782	**0.384196**	0.438384	0.438	0.408081
Face (all)	**0.101333**	0.106667	0.286391	0.286	0.193491

performance in terms of classification accuracy and time consuming.

4.3. UCR Time Series Classification. Besides the above two data sets, we also test our Cycle_DBN on the ten distinct time series datasets from UCR time series [20]. All the dataset has been split into training and testing by default. The only preprocessing in our experiment is normalization and divides them into training, validating, and testing set.

Table 4 shows the test error rate and a comprehensive comparison with KNN_ED, KNN_DTW, RNN, DBN, and Cycle_DBN.

Cycle_DBN outperforms other four methods on five datasets of ten datasets; KNN_ED and KNN_DTW achieve best performance on the same two datasets. DBN achieves best performance on two datasets. Although the performance of RNN is not prominent, the effect is also acceptable.

5. Conclusion

Time series classification is becoming more and more important in a broad range of real-world applications. However, most existing methods have lower classification accuracy or need domain knowledge to identify representative features in data. In this paper, we proposed a Cycle_DBN for classification of multivariate time series data in general. Like DBN, Cycle_DBN is an unsupervised learning algorithm which can discover the structure hidden in the data and learn representations that are more suitable as input to a supervised machine than the raw input. Comparing with DBN, the new model Cycle_DBN predicts the label of time t y_t not only based on the current input x_t but also based on the label of previous time y_{t-1}. We evaluated our Cycle_DBN model on twelve real-world datasets and experimental results show that our model outperforms DBN, KNN_ED, KNN_DTW, and RNN on most datasets.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 51574232, 61379100, 61673196, and 61502212).

References

[1] L. Chen and M. S. Kamel, "Design of multiple classifier systems for time series data," *Lecture Notes in Computer Science*, pp. 216–225, 2005.

[2] I. Batal et al., "Multivariate time series classification with temporal abstractions," *The Florida Ai Research Society*, vol. 22, 344 pages, 2009.

[3] J. J. Rodriguez, C. J. Alonso, and H. Bostrom, "Boosting interval based literals," *Intelligent Data Analysis*, vol. 5, no. 2, pp. 245–262, 2001.

[4] J. J. Rodríguez and C. J. Alonso, *Interval and Dynamic Time Warping-Based Decision Trees*, p. 548, 2004.

[5] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," in *Information Processing and Technology*, M. Nikos and D. N. Stavros, Eds., pp. 49–61, Nova Science Publishers, Inc., New York, USA, 2001.

[6] S. Kim, P. Smyth, and S. Luther, "Modeling waveform shapes with random effects segmental hidden Markov models," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pp. 309–316, AUAI Press, Banff, Canada, July 2004.

[7] D. Zhang, W. Zuo, D. Zhang, and H. Zhang, "Time series classification using support vector machine with Gaussian elastic metric kernel," in *Proceedings of the 2010 20th International Conference on Pattern Recognition (ICPR '10)*, IEEE, Istanbul, Turkey, August 2010.

[8] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, 2003.

[9] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the the 23rd International Conference on Machine learning (ICML '06)*, pp. 1033–1040, ACM, Pittsburgh, Pensylvania, USA, June 2006.

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[11] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*, pp. 609–616, ACM, Montreal, Quebec, Canada, June 2009.

[12] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] M. W. Amp, M. Rosenzvi, and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS '04)*, 2004.

[14] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Proceedings of the International Conference on Neural Information Processing*, 2006.

[15] J. Chao, F. Shen, and J. Zhao, "Forecasting exchange rate with deep belief networks," in *Proceedings of the 2011 International Joint Conference on Neural Network (IJCNN '11)*, IEEE, San Jose, CA, USA, August 2011.

[16] M. Längkvist, L. Karlsson, and A. Loutfi, "Sleep stage classification using unsupervised feature learning," *Advances in Artificial Neural Systems*, vol. 2012, Article ID 107046, 9 pages, 2012.

[17] J. Kruskall and M. Liberman, "Thesymmetric time warping problem: from continuous to discrete," in *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence-Comparison*, Addison-Wesley, 1983.

[18] G. A. ten Holt, M. J. Reinders, and E. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in *Proceedings of the Thirteenth annual conference of the Advanced School for Computing and Imaging*, 2007.

[19] A. Rechtschaffen and A. Kales, *A Manual of Standardized Terminology, Techniques, and Scoring Systems for Sleep Stages of Human Subjects*, 1968.

[20] Y. Chen et al., The ucr time series classification archive, http://www.cs.ucr.edu/~eamonn/time_series_data/, 2015.