Sensors and Materials, Vol. 33, No. 4 (2021) 1149–1165
MYU Tokyo

S & M 2524

Genetic-algorithm-based Convolutional Neural Network for Robust Time Series Classification with Unreliable Data

Jiang Wu,[1*] Yanju Ji,[2] and Suyi Li[2]

[1]Changchun Institute of Technology, No. 395 Kuanping Road, Changchun City 130012, China
[2]College of Instrumentation and Electrical Engineering, Jilin University,
No. 2699 Qianjin Street, Changchun City 130061, China

(Received September 24, 2020; accepted March 8, 2021)

Keywords: genetic algorithm, convolutional neural network, time series classification, photoplethysmography

Finding robust solutions to time series classification problems using deep neural networks has received wide attention. However, unreliable data makes classification very difficult. Traditional deep neural networks cannot effectively solve problems with strong noise. In this paper, we propose a hybrid convolutional neural network (CNN) model combined with a genetic algorithm (GA) for time series classification (TSC) with unreliable data. To obtain a robust CNN structure, even though network structural optimization is an NP-hard problem, we design a GA for network structure optimization. Several benchmarks and actual datasets are adopted, and tests are carried out to prove the effectiveness of the proposed GA-based CNN. The numerical results show that our approach has better performance than other state-of-the-art deep neural networks.

1. Introduction

Time series classification (TSC) is one of the most important problems in machine learning and data mining. The target of TSC is to discover a classification model that identifies the data characteristics, where the data are a description of a series of datasets indexed in time order. TSC problems arise in a wide range of fields including natural language processing, image processing, scheduling, logistics, medicine, and health. For more extensive explanations of the various TSC problems, the reader is referred to previous reviews.[1,2]

The research on TSC has been ongoing for decades. There are three categories of approaches for TSC, i.e., distance-based classification, feature-based classification, and support vector machine (SVM) and model-based classification.[3] For distance-based classification, Faloutsos *et al.* proposed a Euclidean-distance-based approach, which is a predefined similarity measure for TSC.[4] Yi and Faloutsos discussed various distance-based measurement strategies and improved the Euclidean distance measurement strategy with common Lp-norms to form extensions of the Euclidean distance.[5] Frentzos *et al.* proposed a dissimilarity metric (DISSIM) to measure the spatiotemporal dissimilarity between two similar time series.[6] The above approaches are lock-step measures. Although they are widely adopted measurements, Berndt

*Corresponding author: e-mail: 183342462@qq.com
https://doi.org/10.18494/SAM.2021.3002

ISSN 0914-4935 © MYU K.K.

and Clifford[7] and Keogh and Ratanamahatana thought that the Euclidean-distance-based measurement strategy and its extensions were insufficiently robust as similarity measures. Thus, they proposed dynamic time warping (DTW), a classic speech recognition tool, which provides a better match with another time series through the compression of the comparable time series. Chen *et al.* presented an edit distance with real sequence (EDR), in which a threshold parameter was adopted, where the distance was quantified according to a threshold given in advance.[9] Chen and Ng proposed an edit distance with real penalty (ERP), in which DTW and EDR were combined and a constant reference point was set to compute the distance.[10] Vlachos *et al.* studied the longest common subsequence (LCSS), which provided a strategy to consider constraining the matching of two points.[11] Similarly, Morse and Patel developed a sequence-weighted alignment model (Swale) for TSC.[12] DWT, EDR, ERP, LCSS, and Swale are examples of elastic measures. In addition to lock-step measures and elastic measures, there are also threshold-based measures, i.e., threshold queries (TQuEST)[13] and pattern-based measures, such as the spatial assembling distance (SpADe).[14] A summary of distance-based measures is listed in Table 1. Although various variants have been researched, they were still distance-based measures or editing distance measures. The measures may work well for simple TSC with low-dimension data; however, they still have difficulty with complex TSC.

For feature-based classification, some basic approaches, such as decision trees and neural networks, and some approaches for feature selection have been studied and adopted to classify feature vectors. Then, the sequence classification of TSC is solved by transforming the sequence through the results of feature selections.[15] Chuzhanova *et al.* proposed a gamma-test-based feature selection for the sequence classification problem.[16] Ji *et al.* studied an approach using contract sequences with a gap miner strategy to mine the distinguishing subsequence satisfying the constraints.[17] Nanopoulos *et al.* presented a feature selection approach for TSC with the help of a multilayer perceptron (MLP) neural network.[18] Yoon *et al.* studied a novel unsupervised method based on the common principal component analysis strategy to select suitable features.[19] The key factor of feature selection is the criteria used to select the features. Eads *et al.* stressed that the most difficult and important part of feature selection is selecting appropriate features. There is necessarily always a trade-off between manual selection and the help of domain experts.[20] Moreover, it is difficult to design good features to capture intrinsic

Table 1
Summary of distance-based measures.

Measure type		Measure name
Lock-step measures	L_p-norms	L_1-norms (Manhattan distance)
		L_2-norms (Euclidean distance)
		L_{int}-norms
	DISSIM	
Elastic measures	DTW	
	Edit-distance-based measure	LCSS
		EDR
		Swale
		ERP
Threshold-based measures	TQuEST	
Pattern-based measures	SpADe	

properties embedded in various time series data. Therefore, the accuracy of feature-based methods is usually lower than that of sequence-distance-based ones, particularly 1-nearest neighbor (1-NN) with DTW. On the other hand, 1-NN and DTW have been used in many studies, but both require too much computation for many real-world applications.[7]

In fact, the techniques mentioned above usually depend on handcrafted features that require researchers to have sufficient professional knowledge and practical experience. Furthermore, even if they need a lot of time and labor, there may still be inevitable deviation during classification. Thus, various researchers have focused on effective approaches based on data mining tools in recent years.

As one of the supervised learning models, SVM is a kind of generalized linear classifier for binary data classification. In recent years, many variants of basic SVM have been applied for TSC. Kampouraki *et al.* investigated the potential benefit of Gaussian kernel-based SVM on heartbeat TSC.[21] Eads *et al.* proposed an algorithm called Zeus for TSC, which employed evolutionary computation for feature extraction and SVM for classification.[22] Alalshekmubarak and Smith changed the output layer of SVM by replacing the linear readout function with the radial basis function kernel, and proposed a novel algorithm that combines SVM and an echo state network for TSC.[23] Rodríguez and Alonso attempted to combine SVM and a boosting algorithm to analyze interval features for time series classification.[24] On the basis of a temporal extension of discrete SVMs, Orsenigo and Vercellis proposed a new algorithm with the benefit of a warping distance and a softened variable margin.[25] Although SVM has strong ability and flexibility in data mining for various applications, it is difficult to interpret the results and difficult for users to gain knowledge other than the classification result, especially for kernel-based methods.

With the increased availability of time series data, the effectiveness of TSC faces enormous challenges. Recently, deep learning (DL) has successfully been applied in various classification tasks. This is because DL can learn a hierarchical feature representation from data automatically instead of preparing the features manually. The following are some typical DL approaches for TSC.

The MLP is a common feedforward artificial neural network model. It maps multiple input datasets to a single output dataset and adjusts parameters through an error back-propagation algorithm. Fawaz *et al.* studied several state-of-the-art DL algorithms for TSC and proposed an open-source DL framework for the TSC community.[26] Zheng *et al.* employed a DL framework to improve feature learning techniques to solve multivariate time series classification.[27] Nanopoulos *et al.* constructed an improved approach based on the MLP for multivariate time series.[28] Batres-Estrada applied a DL framework to multivariate financial time series and demonstrated the effectiveness of the MLP in TSC.[29]

A recurrent neural network (RNN) can describe dynamic time behavior because, unlike feedforward neural networks that accept input from more specific structures, RNNs pass states in their own networks, thus accepting a wider range of time series structure inputs. The main purpose of an RNN is to process and predict sequence data. It is used to model sequence data, which means that the current output of a sequence is also related to the previous output. Because of its network structure, an RNN will remember previous information and use it to affect the output of the following nodes.

Although RNNs have many advantages, there is an obvious problem: long-term dependence. As a kind of RNN, long short-term memory (LSTM) can learn long-term dependence information and has a gate mechanism to control the flow and loss of features to avoid the long-term dependence problem. Since LSTM was proposed by Hochreiter and Schmidhuber in 1997,[30] many researchers have contributed to the modern LSTM, such as Felix Gers, Fred Cummins, and so on, and a complete system for LSTM has been formed. Owing to the unique design structure of LSTM, it is suitable for processing and predicting important events with very long intervals and delays in time series. Lipton *et al.* augmented fully convolutional networks (FCNs) with LSTM-RNN submodules for TSC, which have been demonstrated to have state-of-the-art performance.[31] Karim *et al.* proposed that LSTM-FCNs and attention LSTM-FCNs (ALSTM-FCNs) should be transformed into a multivariate TSC model by augmenting a fully convolutional block.[32] Lipton *et al.* applied LSTM-RNNs to the multilabel classification of diagnoses and demonstrated their effectiveness.[33] Malhotra *et al.* proposed TimeNet based on a deep RNN, which can be trained on diverse time series in an unsupervised manner.[34]

XGBoost is an open-source software library that provides a gradient-boosting framework. From the project description, it aims to provide a "scalable, portable and distributed gradient boosting (GBM, GBRT, GBDT) library." In addition to running on a single machine, it also supports distributed processing frameworks such as Apache Hadoop, Apache Spark, and Apache Flink. It has gained much popularity and attention recently as it was the algorithm of choice for many winning teams of a number of machine learning competitions. Zheng *et al.* proposed a short-term load forecasting method using EMD-LSTM neural networks with an XGBoost algorithm for feature importance evaluation.[35] Chen *et al.* proposed a radar emitter classification for large datasets based on weighted XGBoost.[36]

The convolutional neural network (CNN) is a variant of the MLP, which was developed by biologists Huber and Wiesel in their early research on the cat visual cortex. The first CNN, LeNet-5, was proposed by Lecun and Bottou in 1998.[37] As a kind of feedforward neural network with a depth structure, a CNN contains a convolution calculation and is one of the representative algorithms of DL. Morabito *et al.* generated suitable sets of features with the help of the representational power of a CNN.[38] Zheng *et al.* proposed a novel DL framework for multivariate TSC.[39] Yang *et al.* proposed a systematic feature learning method for the HAR problem that adopts a deep CNN to automate feature learning from the raw inputs in a systematic way.[40] Researchers found that the cooperation and combination of a CNN and LSTM can have good performance. Zhou *et al.* proposed a combination of a CNN and LSTM for text classification.[41] Wang *et al.* proposed a beyond frame-level CNN, which is a saliency-aware 3D CNN with LSTM for recognition.[42] Wu and Prasad proposed a novel algorithm based on a CNN with the help of LSTM for hyperspectral data classification.[43]

However, time series datasets are often mixed with strong noise. For example, noisy environments will reduce the effect of natural language processing, and water droplets falling on a camera on rainy days can reduce the accuracy of video detection. At present, the design of network structures is mostly oriented to the kind of problem instead of the kind of dataset. As a result, the performance of a network may vary greatly on different datasets for the same kind of problem. To solve this problem, we attempt to adjust the network structure according to

the characteristics of the dataset in the training process to make up for the lack of a traditional learning model. In this paper, we combine a genetic algorithm (GA) with a CNN and propose a hybrid model (GACNN), in which the CNN is trained for a certain number of epochs, and then its structure is adjusted by the GA.

The rest of the paper is organized as follows. In Sect. 2, we present the problem description and the model of TSC. Our proposed GACNN is introduced in Sect. 3. Section 4 presents numerical experiments, and Sect. 5 presents the final conclusion.

2. Description of Problem

TSC can be defined as a classification problem with a series of datasets indexed in time order. Three kinds of time series are defined:[2]

- A single time series: $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ is a vector with real values, where n is the number of real values.
- A multidimensional time series: $X = [\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^i, ..., \boldsymbol{x}^m]$ is a matrix with m different single time series, where $\boldsymbol{x}^i$ is the ith single time series.
- A dataset of time series: $D = \{(X_1, \boldsymbol{y}_1), (X_2, \boldsymbol{y}_2), ..., (X_j, \boldsymbol{y}_j), ..., (X_N, \boldsymbol{y}_N)\}$ is a pair dataset $(X_j, \boldsymbol{y}_j)$, where X_j is a multidimensional time series (or a single time series) and $\boldsymbol{y}_j$ is the corresponding label vector.

We consider a single time series $\boldsymbol{x} = [x_i]$. A time series $\boldsymbol{x}$ of length n is split into fixed-size windows $S_{i:w} = [x_i, ..., x_{i+w-1}]$ of length w using a windowing function. Two consecutive windows at offsets i and $i + 1$ overlap at $w - 1$ positions:

$$win(\boldsymbol{x}, w) = \left\{ \underbrace{S_{1:w}}_{[t_1,...,t_w]}, \underbrace{S_{2:w}}_{[t_2,...,t_{w+1}]}, ..., S_{n-w+1:w} \right\}. \tag{1}$$

In the training phase, the objective of TSC is to find the correspondence between windows $\{S_{i:w}\}$ and K feature classes using a learning model. Here, we set a probability distribution over K classes with each label value of $\boldsymbol{y}_j \in [1, K]$, $j = 1, 2, ..., K$. An illustration of TSC is shown in Fig. 1. Given a sequence of values for a time series dataset D, values at multiple time steps

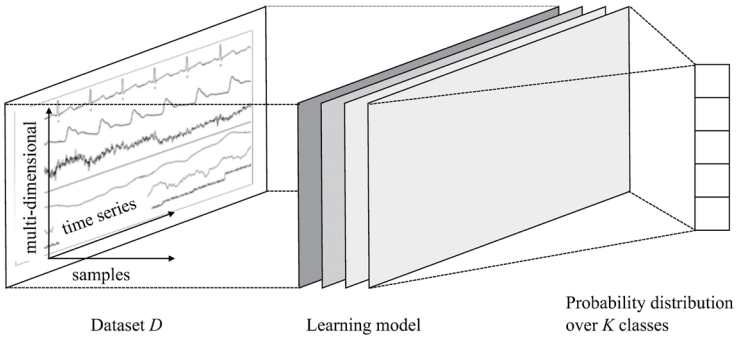


Fig. 1. Illustration of time series classification.

can be grouped to form an input vector (generally provided by an expert). Algorithm 1 shows a general implementation process of a TSC learning algorithm.

Algorithm 1: TSC learning algorithm.

1. **Input**: time series dataset D, windows $\{S_{i:w}\}$, probability distribution $\{y_j\} \in K$ classes
2. Output: learning model $f(\theta)$
3. **Begin**
4. $i = 0$;
5. **while** $i+w \leq length(D)$ **do** // w num. of window size
6. $T_{i:w} = D_{i:i+w}$; // $T_{i:w}$ time series sequence
7. **update** $\theta = learning(\{S\}, f(\theta, T_{i:w}))$;
8. **end**
9. **return** $f(\theta)$
10. **End**

Example of TSC: Sleep apnea syndrome (SAS) is a sleep disorder that seriously affects sleep quality and threatens public health. Polysomnography (PSG) signals, the diagnostic gold standard for SAS, are a kind of multidimensional time series: a collection of recordings of multiple physiologic signals during sleep. The dataset of PSG signals includes electrocardiogram (ECG), photoplethysmography (PPG), electroencephalogram (EEG), stroke volume, and peripheral oxygen (SpO2) signals. A sample dataset of PSG is shown in Fig. 2.

In general, a doctor can judge the patient's sleep staging and apnea condition on the basis of the PSG signals. There are six types of apnea conditions as follows:

H Hypopnea
HA Hypopnea with arousal
OA Obstructive apnea
X Obstructive apnea with arousal
CA Central apnea
CAA Central apnea with arousal

Solving TSC has great significance, such as the above SAS classification by PSG. However, in actual situations, the sampling data are mixed with strong noise. This makes the learning of data characteristics and the mining of features from sampling data too difficult. As shown in Fig. 3, there are significant differences in the data in different situations. Therefore, the learning model

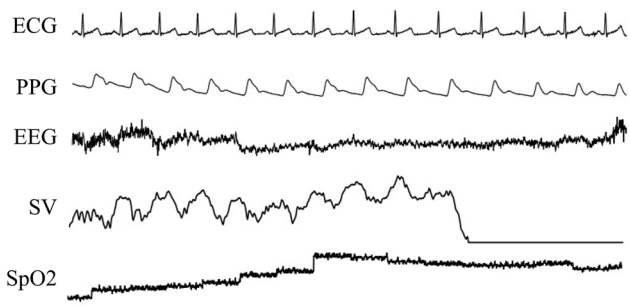


Fig. 2. Sample dataset of multiple physiologic signals during sleep.

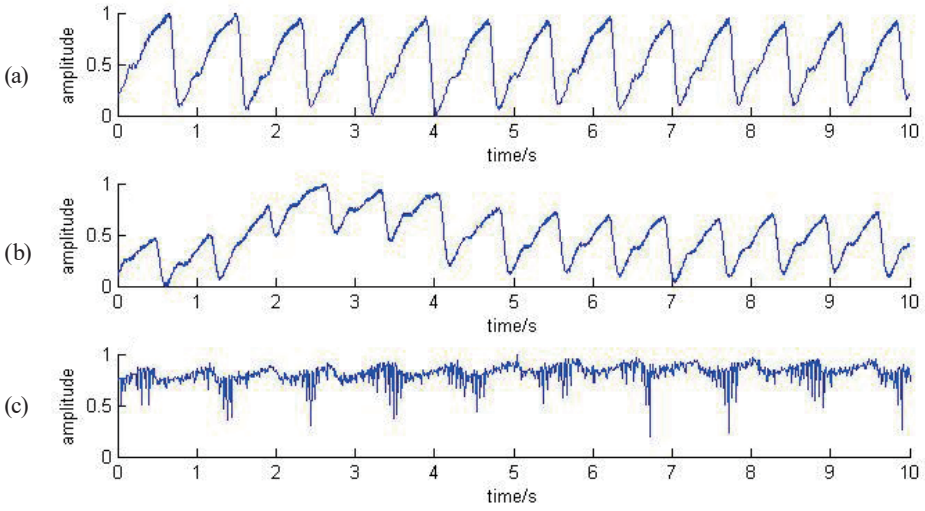


Fig. 3. (Color online) Measured PPG signals. (a) Raw signal when sitting calmly; (b) raw signal when moving slightly; (c) raw signal acquired in strongly noisy environment.

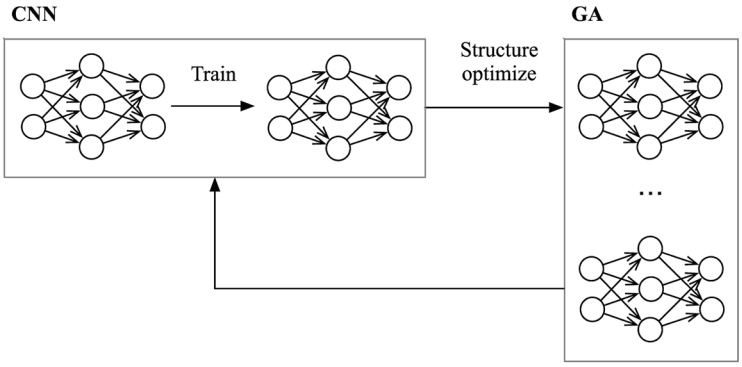


Fig. 4. Algorithm flowchart of GACNN.

must be robust to unreliable data. In this paper, we propose a hybrid CNN model combined with a GA for TSC with unreliable data. To obtain a robust CNN network structure, we design a GA for network structure optimization.

3. GACNN

A GACNN aims to find a robust DL model to fit unreliable data. In this paper, the GACNN focuses on TSC problems, where the TSC data are mixed with strong noise. The GACNN considers both the CNN and GA as its basic algorithms and is suitable for TSC and network structure optimization. First, we design and train a full CNN using TSC sampling data. After a certain number of training steps, we adopt the GA to adjust the CNN structure by cutting some connections between neurons. By repeating the above processes alternately, the GACNN can obtain a more efficient DL model. A flowchart for GACNN is shown in Fig. 4, and its overall

algorithm is shown in Algorithm 2. All details regarding the GACNN are introduced separately in the following.

Algorithm 2: GACNN.

```
1. Begin
2.     Initialize a CNN;
3.     Train the CNN for TSC;
4.     for CNN structure optimization step do
5.         GA-based CNN structure optimization;
6.         fine-tune the remaining CNN;
7.     end for
8.     return fine-tune the remaining CNN;
9. End
```

3.1 CNN for TSC

A deep neural network is a composition of L layers of a bipartite graph with weighted and directed arcs. Each layer l_i, $i \in (1, L)$ contains neurons, takes the output of the previous layer l_{i-1} as the input, and applies an activation function to compute its output:

$$f_L(\theta_L, \boldsymbol{x}) = f_{L-1}(\theta_{L-1}, f_{L-2}(\theta_{L-2}, f_{L-3}(\theta_{L-3}, ..., f_1(\theta_1, \boldsymbol{x})))), \tag{2}$$

where f_i corresponds to the activation function applied at layer l_i using weighted parameters θ_i and input values $\boldsymbol{x}$.

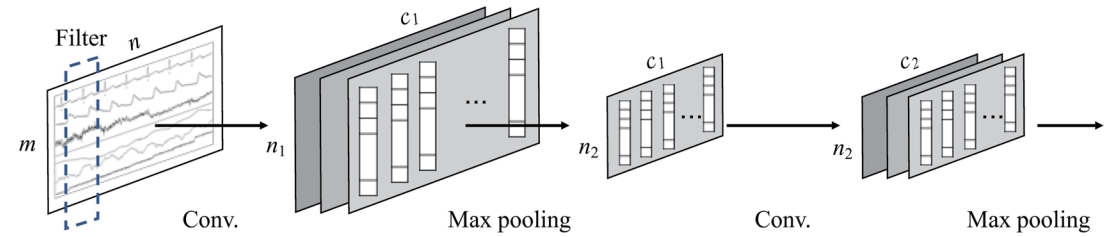
The CNN is one of the classical deep neural networks and is most commonly applied to computer vision. Recently, CNNs have been successfully applied in various fields, including TSC.[1] To solve TSC problems, the convolution is defined as applying and sliding a filter over the time series. The filter of CNN expresses time series as one dimension or multiple dimensions. A general function applying the convolution for a time stamp t is given as

$$C_t = f(\theta, X_t^F + \boldsymbol{b}),\ \forall t \in [1, N], \tag{3}$$

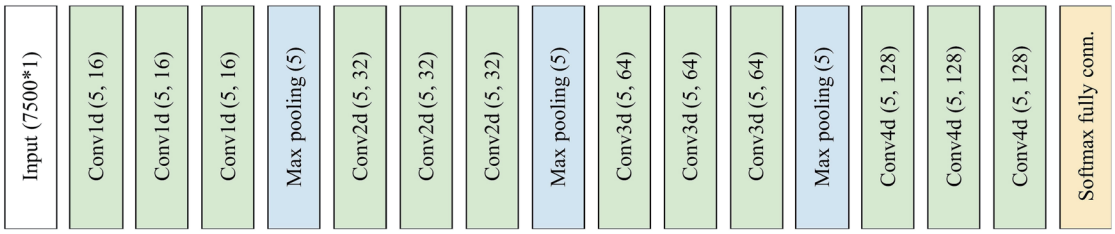
where C_t denotes the convolution applied to the dataset of time series $D = \{(X_t, \boldsymbol{y}_t)\}$ on time stamp t. X_t^F is calculated from X_t using a filter F. An example of a CNN structure for TSC with four convolutional layers is illustrated in Fig. 5.

3.2 GA-based CNN structure optimization

GAs have attracted wide attention because of their intelligence, parallelism, robustness, good adaptability, and the capability of global searching. A GA is a generic population-based meta-heuristic optimization algorithm that uses some mechanisms inspired by biological evolution: mutation, crossover, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions 'live'.



(a)



(b)

Fig. 5. (Color online) CNN for time series classification. (a) Illustration of Conv. and Max pooling. (b) Example of CNN structure for SAS classification.

A CNN structure can be represented by a special directed graph. Consider a CNN $G = (N, W)$ consisting of a finite set of neurons $N = \{n_i\}$ and a set of weights $W = \{w_k \mid (n_i, n_j)\}$ joining pairs of neurons in N. For a given dataset of time series $\boldsymbol{X} = \{X_i\}$, the loss function of TSC can formulated as follows:

$$\mathbb{E}(\boldsymbol{X}, G). \tag{4}$$

An optimized CNN structure G' can be obtained as $(N, W{\cdot}U)$, where $U = \{u_k\}$, $u_k \in (0, 1)$ is a transition matrix. The structure optimization can be formulated as follows:

$$\min \left\| \mathbb{E}(X, N, W) - \mathbb{E}(X, N, W \cdot U) \right\| + \lambda \Omega(U), \tag{5}$$

where $\|\cdot\|$ represents the loss between the original structure G and the optimized structure G', and $\lambda\Omega(U)$ is the penalty function of the transition matrix U. Formally, the objective function used in CNN structure optimization can be formulated as follows:

$$\arg\min_{U} \left\| \mathbb{E}(X, N, W) - \mathbb{E}(X, N, W \cdot U) \right\| + \lambda \Omega(U). \tag{6}$$

In the above description, this structure optimization has a computational complexity of $O(2^n)$ and is an NP-hard problem. The procedure of CNN structure optimization is shown in Algorithm 3.

Algorithm 3: CNN structure optimization.

1. **Initialize** model $G(N, W)$;
2. **while** not end of training **do**
3. **for** setting steps **do**
4. training by $\mathbb{E}(\boldsymbol{X}, G)$;
5. **end for**
6. evaluate model $G(N, W)$;
7. optimize network structure $W' \leftarrow W \cdot U$ via GA;
8. update $G(N, W')$
9. **end while**

Representation

How to encode a CNN structure into a chromosome is a key issue of GAs. In this paper, the main objective is to determine the optimal combination of $\{u_k \in (0, 1)\}$, and the representation is carried out using binary strings. A gene $v_k(w_{ij})$ can be seen as a switch that controls the existence of a connection from node i to node j. If the state of $v_k(w_{ij})$ is 1, then neuron i is connected with neuron j in the CNN, and if the state of $v_k(w_{ij})$ is 0, then the connection between neuron i and neuron j is deleted from the CNN. An example of a representation is shown in Fig. 6 and a decoded result is shown in Fig. 7.

Evolution

To search for an effective network structure, we evaluate a representation by the following evolution function:

$$fitness = \alpha \left\| \mathbb{E}(X, N, W) - \mathbb{E}(X, N, W \cdot U) \right\| + \lambda \Omega(U) \tag{7}$$

$$\Omega(U) = 1 - \frac{\sum_{k=1}^{N} u_k}{N}, \tag{8}$$

where α and λ are adaptive weight parameters with adjustable weights, in order to adjust the search direction between the loss function and network structure.

locus:	w_{13}	w_{14}	w_{15}	w_{23}	w_{24}	w_{25}	w_{36}	w_{37}	w_{46}	w_{47}	w_{56}	w_{57}
U:	1	1	1	0	1	0	0	1	1	0	1	1

Fig. 6. Binary string for CNN structure representation.

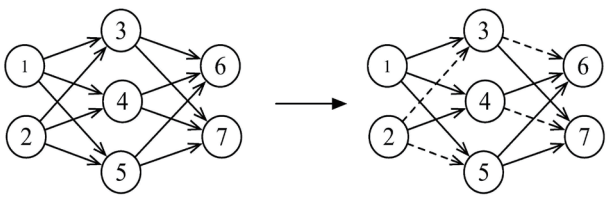


Fig. 7. Network structure adjustment for representation in Fig. 6.

$$z_1 = \left\| \mathbb{E}(X, N, W) - \mathbb{E}(X, N, W \cdot U) \right\| \tag{9}$$

$$z_2 = \Omega(U) \tag{10}$$

The weights are defined as follows, where r is a nonnegative random number in the interval [0, 1].

$$\alpha = \frac{r}{z_1^{max} - z_1^{min}} \tag{11}$$

$$\lambda = \frac{1 - r}{z_2^{max} - z_2^{min}} \tag{12}$$

Mutation

Mutation is a genetic operator that produces spontaneous random changes in various chromosomes. In the GA, mutation serves the crucial role of either (a) replacing the genes lost from the population during the selection process so that they can be tried in a new context or (b) providing genes that were not present in the initial population. Mutation selects a gene at random; since the gene is 1 (or 0), it will be flipped to 0 (or 1). An example of mutation is shown in Fig. 8.

Overall procedure of GA

The implementation procedure of the GA is described in Algorithm 4, where $P(t)$ and $C(t)$ are parents and offspring, respectively, in the current generation t.

Algorithm 4: GA.

```
1.  initialize P(t) by random generated structure U;
2.  fitness eval(P) by evolution function;
3.  while not termination condition do
4.      make P(t) to popsize/2 pairs;
5.      for each pair do
6.          compare fitness;
7.          select the better individual as offspring C(t);
8.          generate offspring C(t) by mutation;
9.      end for
10.     fitness eval(C) by evolution function;
11. end while
12. output the best individual;
```

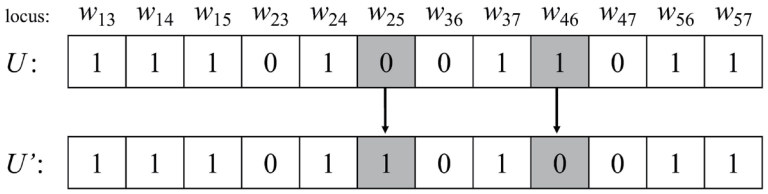


Fig. 8. Example of mutation.

4. Experiments

We apply our GACNN to the MIT-BIH polysomnographic database[44,45] and measured PPG signals, and compare its performance with six state-of-the-art models: SVM, MLP, LSTM, CNN, CNN+LSTM, and XGBoost. All algorithms are implemented on TensorFlow 1.10 with cuDNN 7.0.

4.1 Datasets

The MIT-BIH polysomnographic database is provided by PhysioNet (www.physionet.org). It is a dataset of multiple physiologic signals during sleep. The dataset was recorded in Boston's Beth Israel Hospital Sleep Laboratory, where subjects were monitored to evaluate SAS and test the effectiveness of constant positive airway pressure. The dataset contains 4/6/7-channel polysomnographic recordings with ECG signals annotated beat-by-beat, and EEG and respiration signals annotated with respect to sleep stages and apnea. The dataset consists of 18 records with 16 male subjects, aged 32–56 years (avg. age 43 years) weighing 89–152 kg (avg. weight 119 kg). In this paper, the sampling frequency of the selected pulse wave signal is 250 Hz. The annotation period of the apnea signal is per 30 s, there are a total of 9602 labels, and the data dimensions are 9602 × 7500.

MIT-BIH with noise synthesis is considered to simulate uncertainty in signal acquisition. We synthesize the dataset slp67x with noise, and the sampling frequency is 250 Hz. A set of reference signals is shown in Fig. 9(a), which is recorded as 10 s data strings starting from 1 h 17 m. First, we superimpose low-frequency mixed sine and cosine signals on the reference signals to simulate baseline drift noise, called MIT-BIH-dn [Fig. 9(b)]; then we superimpose white Gaussian noise with a 10 dB signal-to-noise ratio and power line interference on the reference

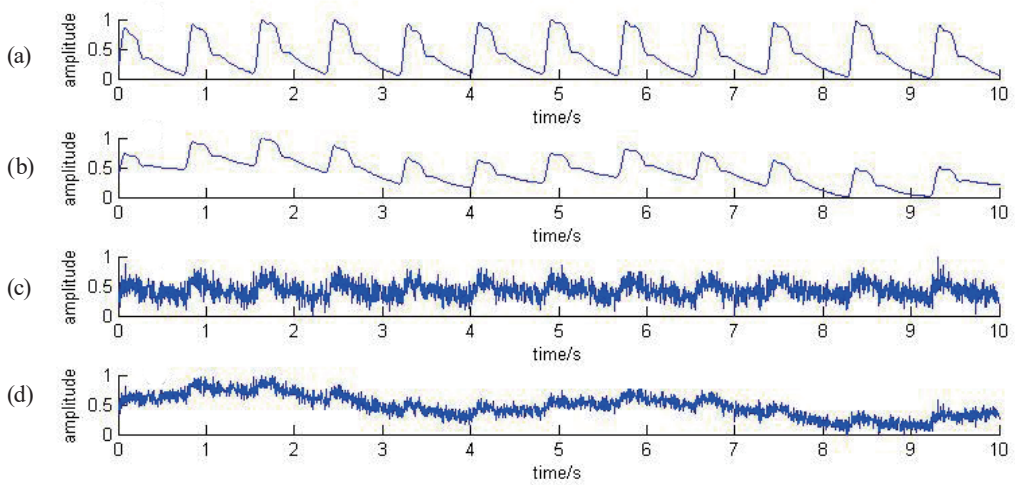


Fig. 9. (Color online) Sample dataset with noise synthesis. (a) Reference signal, (b) synthetic signal with added baseline drift noise, (c) synthetic signal with added random noise and power line interference, and (d) synthetic signal with added multiple noises.

signals, called MIT-BIH-rn [Fig. 9(c)]; lastly, we superimpose multiple noises combined with MIT-BIH-dn and MIT-BIH-rn, called MIT-BIH-mn [Fig. 9(d)].

Actual PPG data are recorded from seven males and three females with age 26.20 ± 5.14 (mean ± std) and BMI 21.79 ± 3.40 (mean ± std). We record multiple sets of PPG signals, including a raw signal with the subject sitting calmly (PPG), a raw signal with the subject moving slightly (PPG-ms), a raw signal acquired under strong noise (PPG-sn), and a raw signal with multiple noises (PPG-mn).

4.2 Experimental settings

For CNN and MLP, training is done using mini-batch SGD with a momentum of 0.9, and the batch size is set to 64 for all the networks. Networks are trained for a total of 100 epochs; we start from a learning rate of 0.01 and divide by two every 10 epochs. For MLP, a dropout is applied after every hidden layer, and the dropout rate is set to 0.2.

For LSTM and CNN-LSTM, we train the network using Adam with a batch size of 64. Both networks are trained for a total of 50 epochs. For SVM and XGBoost, we directly use the raw time series samples as the input. SVM with a linear kernel is used as the classifier and the penalty C is set to 0.4. For XGBoost, the maximum depth is set to 20 and the maximum number of iterations is set to 30. For GACNN, the training schema is similar to that of CNN and MLP, using mini-batch SGD with a momentum of 0.9. We apply a GA search at the 30th epoch to adapt the structure of the network once. The population size is set to 50 with evolution for 100 generations.

4.3 Results on 2-type classifiers

We perform experiments focusing on 2-type classifiers with MIT-BIH benchmarks and PPG actual datasets. Firstly, we execute GACNN and six other algorithms on two conventional datasets: MIT-BIN 2-classifier (results shown in Table 2) and PPG 2-classifier (results shown in Table 3). The CNN-based algorithms (CNN, CNN+LSTM, and GACNN) achieve better performance than SVM, MLP, LSTM, and XGBoost, except for CNN+LSTM, which has lower performance than MLP on the MIT-BIH 2-classifier. Our GACNN achieves the best performance. Furthermore, after optimizing the CNN network structure, about 66% and 44% of connections are pruned using the GA, thus increasing the effectiveness of the algorithm.

Table 2
MIT-BIH 2-classifier.

Method	Test error	Parameter	CTS (ms)
SVM	46.38%	—	24.14
MLP	30.62%	8002502	0.32
LSTM	33.14%	379	932.63
XGBoost	35.40%	—	0.28
CNN	29.79%	300642	3.82
CNN+LSTM	31.86%	10486	25.81
GACNN	**28.33%**	103380	2.18

Table 3
PPG 2-classifier.

Method	Test error	Parameter	CTS (ms)
SVM	31.70%	—	43.26
MLP	24.31%	4252502	0.22
LSTM	26.45%	162	621.75
XGBoost	28.79%	—	0.14
CNN	21.38%	275554	0.88
CNN+LSTM	21.26%	9778	22.72
GACNN	**20.66%**	155040	0.79

Table 4
MIT-BIH 2-classifier with noise.

Method	MIT-BIH-dn	MIT-BIH-rn	MIT-BIH-mn
SVM	44.19%	48.76%	46.68%
MLP	33.71%	37.28%	34.84%
XGBoost	35.89%	37.97%	37.97%
CNN	30.34%	31.26%	32.79%
CNN+LSTM	33.42%	32.75%	32.92%
GACNN	**28.75%**	**30.75%**	**31.25%**

Table 5
PPG 2-classifier with noise.

Method	PPG-ms	PPG-sn	PPG-mn
SVM	43.61%	44.61%	44.29%
MLP	26.54%	28.71%	27.91%
XGBoost	28.33%	30.83%	31.56%
CNN	22.94%	23.03%	23.32%
CNN+LSTM	24.36%	24.63%	25.70%
GACNN	**21.45%**	**22.33%**	**22.73%**

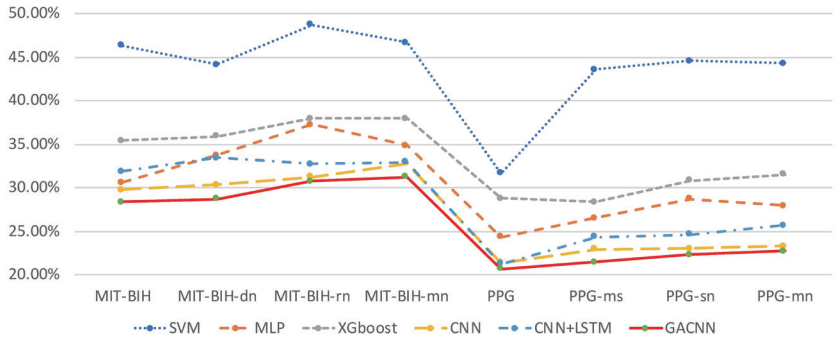


Fig. 10. (Color online) Summary of performance of seven algorithms in eight experiments.

Then, we execute experiments on two unreliable datasets, MIT-BIH 2-classifier with noise (results shown in Table 4) and PPG 2-classifier with noise (results shown in Table 5). Six noise experiments are performed. Again, our GACNN achieves the best performance in all experiments. The results of the experiments are summarized in Fig. 10.

4.4 Results on 7-type classifications

Next, we perform experiments focusing on the 7-type classifications with actual PPG datasets. First, we execute GACNN and five other algorithms on the conventional dataset PPG 7-classifier (results shown in Table 6). Then, three noise experiments are performed to compare the effectiveness of the six algorithms under unreliable data conditions (results shown in Table 7). Our GACNN achieves the best performance in all experiments. The results of the experiments are illustrated in Fig. 11.

Table 6
PPG 7-classifier.

Method	Test error	Parameter	CTS (ms)
SVM	33.11%	—	51.98
MLP	30.83%	4255007	0.22
LSTM	29.23%	187	874.46
XGBoost	31.65%	—	0.27
CNN	23.76%	283239	0.91
CNN+LSTM	26.46%	9943	23.76
GACNN	**22.73%**	118919	0.75

Table 7
PPG 7-classifier with noise.

Method	PPG-ms	PPG-sn	PPG-mn
SVM	45.80%	46.92%	47.26%
MLP	32.01%	35.21%	34.26%
XGBoost	31.33%	31.61%	31.56%
CNN	25.49%	26.03%	26.29%
CNN+LSTM	27.35%	27.03%	29.33%
GACNN	**24.44%**	**25.03%**	**25.62%**

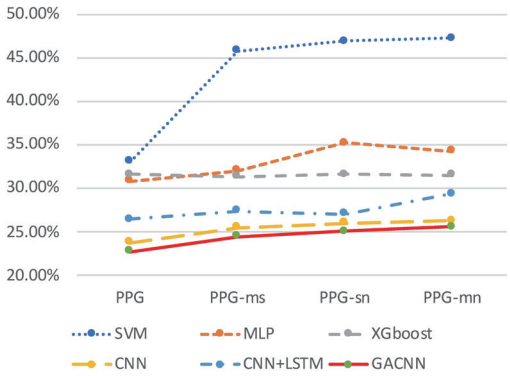


Fig. 11. (Color online) Performance of six algorithms in four experiments.

5. Conclusion

In this paper, we proposed a hybrid CNN model combined with a GA to find robust solutions to time series classification problems. As discussed in the paper, in the CNN training process, the network structural adjustment achieved effective results with robust time series classification. Although this network structural optimization is an NP-hard problem, our structure optimization approach by the GA showed outstanding performance in solving this problem. Benchmarks and actual datasets were adopted and tested to prove the effectiveness of the proposed GACNN. The numerical results showed that our approach had superior performance to six state-of-the-art deep neural networks.

References

1 J. C. B. Gamboa: Deep Learning for Time-series Analysis, arXiv preprint (2017). https://arxiv.org/abs/1701.01887
2 H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller: Data Min. Knowl. Discovery **33** (2019) 917. https://link.springer.com/article/10.1007/s10618-019-00619-1
3 Abanda, Amaia, U. Mori, and J.A. Lozano: Data Min. Knowl. Discovery **33** (2019) 378.
4 C. Faloutsos, M. Ranganathan, and Y. Manolopoulos: Fast: ACM (1994).
5 B. K. Yi and C. Faloutsos: Prof. 26th Int. Conf. Very Large Data Bases **99** (2000) 385. http://www.vldb.org/conf/2000/P385.pdf
6 E. Frentzos, K. Gratsias, and Y. Theodoridis: IEEE Int. Conf. Data Engineering (2007) 816–825. https://doi.org/10.1109/ICDE.2007.367927
7 D. J. Berndt and J. Clifford: 3rd Int. Conf. Knowledge Discovery and Data Mining. AAAI **10.16** (1994) 359–370. https://www.aaai.org/Papers/Workshops/1994/WS-94-03/WS94-03-031.pdf
8 E. Keogh and C. A. Ratanamahatana: Knowl. Inf. Syst. **7** (2005) 358. https://link.springer.com/article/10.1007/s10115-004-0154-9
9 L. Chen, M. T. Özsu, and V. Oria: SIGMOD Int. Conf. Management of Data (ACM, 2005) 491–502. https://dl.acm.org/doi/abs/10.1145/1066157.1066213
10 L. Chen and R. Ng: 30th Int. Conf. Very Large Data Bases (2004) 792–803. https://doi.org/10.1016/B978-012088469-8.50070-X
11 M. Vlachos, G. Kollios, and D. Gunopulos: IEEE 18th Int. Conf. Data Engineering (2002) 673–684. https://doi.org/10.1109/ICDE.2002.994784
12 M. D. Morse and J. M. Patel: SIGMOD Int. Conf. Management of Data (ACM, 2007) 569–580. https://dl.acm.org/doi/abs/10.1145/1247480.1247544
13 J. Aßfalg, H. P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz: Int. Conf. Extending Database Technology **19** (2006) 276. https://doi.org/10.1007/11687238_19

14 Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. Tung: IEEE 23rd Int. Conf. Data Engineering (2007) 786–795. https://ieeexplore.ieee.org/document/4221727
15 Z. Xing, J. Pei, and E. J. Keogh: SIGKDD Explorations Newsletter ACM **12** (2010) 40. https://dl.acm.org/doi/10.1145/1882471.1882478
16 N. A. Chuzhanova, A. J. Jones, and S. Margetts: Bioinformatics **14** (1998) 139. https://doi.org/10.1093/bioinformatics/14.2.139
17 X. Ji, J. Bailey, and G. Dong: Int. Conf. Data Mining (IEEE, 2005) 8. https://ieeexplore.ieee.org/document/1565679
18 A. Nanopoulos, R. Alcock, and Y. Manolopoulos: Feature-based Classification of Time-series Data (2001). https://datalab.csd.auth.gr/wp-content/uploads/publications/IJCR02nam.pdf
19 H. Yoon, K. Yang, and C. Shahabi: IEEE Trans. Knowledge and Data Engineering **17** (2005) 1186. https://doi.org/10.1109/TKDE.2005.144
20 D. Eads, K. Glocer, S. Perkins, and J. Theiler: 9th Annu. Conf. Neural Information Processing Systems (2005) 1–8. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.4333&rep=rep1&type=pdf
21 A. Kampouraki, G. Manis, and C. Nikou: IEEE Trans. Inf. Technol. Biomed. **13** (2008) 512. https://doi.org/10.1109/TITB.2008.2003323
22 D. R. Eads, D. Hill, S. Davis, D. R. Eads, D. Hill, S. Davis, S. J. Perkins, J. Ma, R. B. Porter, and J. P. Theiler: Soc. Opt. Photon. **4787** (2002) 74. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.4333
23 A. Alalshekmubarak, and L. S. Smith: IEEE 9th Int. Conf. Innovations in Information Technology (2013) 42–47. https://doi.org/10.1109/Innovations.2013.6544391
24 J. J. Rodríguez, and C. J. Alonso: Res. Develop. Intell. Syst. XXI. SGAI (2005) 244–245. https://doi.org/10.1007/1-84628-102-4_18
25 C. Orsenigo and C. Vercellis: Pattern Recognit. **43** (2010) 3787. https://doi.org/10.1016/j.patcog.2010.06.005
26 H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller: Data Min. Knowl. Discovery **33** (2019) 917. https://link.springer.com/article/10.1007/s10618-019-00619-1
27 Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao: Int. Conf. Web-age Information Management (2014) 298–310. https://link.springer.com/chapter/10.1007%2F978-3-319-08010-9_33
28 A. Nanopoulos, R. Alcock, and Y. Manolopoulos: Inf. Process. Technol. (2001) 49. https://dl.acm.org/doi/10.5555/766914.766918
29 B. Batres-Estrada: Deep Learning for Multivariate Financial Time Series (2015).
30 S. Hochreiter and J. Schmidhuber: Neural Comput. **9** (1997) 1735. https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735
31 Z. C. Lipton, D. C. Kale, and R. C. Wetzel: Phenotyping of Clinical Time Series with LSTM Recurrent Neural Networks, arXiv preprint (2015). https://arxiv.org/abs/1510.07641
32 F. Karim, S. Majumdar, H. Darabi, and S. Harford: Neural Networks **116** (2019) 237. https://doi.org/10.1016/j.neunet.2019.04.014
33 Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel: Learning to Diagnose with LSTM Recurrent Neural Networks, arXiv preprint (2015). https://arxiv.org/abs/1511.03677
34 P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff: TimeNet: Pre-trained Deep Recurrent Neural Network for Time Series Classification, arXiv preprint (2017). https://arxiv.org/abs/1706.08838
35 H. Zheng, J. Yuan, and L. Chen: Energies **10** (2017) 1168. https://doi.org/10.3390/en10081168
36 W. Chen, K. Fu, J. Zuo, X. Zheng, T. Huang, and W. Ren: IET Radar Sonar Navig. **11** (2017) 1203. https://doi.org/10.1049/iet-rsn.2016.0632
37 Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner: Proc. IEEE **86** (1998) 2278. https://doi.org/10.1109/5.726791
38 F. C. Morabito, M. Campolo, C. Ieracitano, J. M. Ebadi, L. Bonanno, A. Bramanti, S. Desalvo, N. Mammone, and P. Bramanti: IEEE 2nd Int. Forum Research and Technologies for Society and Industry Leveraging a Better Tomorrow (2016) 1–6. https://doi.org/10.1109/RTSI.2016.7740576
39 Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao: Int. Conf. Web-age Information Management (2014) 298–310. https://link.springer.com/chapter/10.1007/978-3-319-08010-9_33
40 J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy: IJCAI **15** (2015) 3995. https://personal.ntu.edu.sg/xlli/publication/HARcnn.pdf
41 C. Zhou, C. Sun, Z. Liu, and F. Lau: A C-LSTM Neural Network for Text Classification, arXiv preprint (2015). https://arxiv.org/abs/1511.08630
42 X. Wang, L. Gao, J. Song, and H. Shen: IEEE Signal Process. Lett. **24** (2016) 510. https://doi.org/10.1109/LSP.2016.2611485
43 H. Wu and S. Prasad: Remote Sens. **9** (2017) 298. https://doi.org/10.3390/rs9030298
44 Y. Ichimaru and G. B. Moody: Psychiatry Clin. Neurosci. **53** (1999) 175. https://doi.org/10.1046/j.1440-1819.1999.00527.x
45 G. B. Moody, R. G. Mark, and A. L. Goldberger: IEEE Eng. Med. Biol. Mag. **20** (2001) 70. https://doi.org/10.1109/51.932728

About the Authors

Jiang Wu is a teacher at Changchun Institute of Engineering, Jilin, China. She received her Ph.D. degree from the College of Instrumentation and Electrical Engineering, Jilin University, China, in 2019. Her research interests include testing technology and automation equipment development. (183342462@qq.com)

Yanju Ji is a professor in the College of Instrumentation and Electrical Engineering, Jilin University, China. She received her Ph.D. degree from Geo-exploration Science and Technology, Jilin University, China, in 2004. Her research interests include the time domain electromagnetic theory and detection technology. (jiyj@jlu.edu.cn)

Suyi Li is a professor in the College of Instrumentation and Electrical Engineering, Jilin University, China. She received her Ph.D. degree in test measurements and instruments from Jilin University, China, in 2009. Her research interests include biomedical signal processing and ocean oil exploration. (lsy@jlu.edu.cn)