SCIENTIFIC REPORTS

OPEN

Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network

Received: 11 January 2019
Accepted: 2 April 2019
Published online: 01 May 2019

Fei Zhu[1,2], Fei Ye[1], Yuchen Fu[3], Quan Liu[1] & Bairong Shen[4]

Heart disease is a malignant threat to human health. Electrocardiogram (ECG) tests are used to help diagnose heart disease by recording the heart's activity. However, automated medical-aided diagnosis with computers usually requires a large volume of labeled clinical data without patients' privacy to train the model, which is an empirical problem that still needs to be solved. To address this problem, we propose a generative adversarial network (GAN), which is composed of a bidirectional long short-term memory(LSTM) and convolutional neural network(CNN), referred as BiLSTM-CNN,to generate synthetic ECG data that agree with existing clinical data so that the features of patients with heart disease can be retained. The model includes a generator and a discriminator, where the generator employs the two layers of the BiLSTM networks and the discriminator is based on convolutional neural networks. The 48 ECG records from individuals of the MIT-BIH database were used to train the model. We compared the performance of our model with two other generative models, the recurrent neural network autoencoder(RNN-AE) and the recurrent neural network variational autoencoder (RNN-VAE). The results showed that the loss function of our model converged to zero the fastest. We also evaluated the loss of the discriminator of GANs with different combinations of generator and discriminator. The results indicated that BiLSTM-CNN GAN could generate ECG data with high morphological similarity to real ECG recordings.

Cardiovascular diseases are the leading cause of death throughout the world. Approximately 32.1% of the annual global deaths reported in 2015 were related with cardiovascular diseases[1]. Due to increases in work stress and psychological issues, the incidences of cardiovascular diseases have kept growing among young people in recent years. As an effective method, Electrocardiogram (ECG) tests, which provide a diagnostic technique for recording the electrophysiological activity of the heart over time through the chest cavity via electrodes placed on the skin[2], have been used to help doctors diagnose heart diseases. However, as vast volumes of ECG data are generated each day and continuously over 24-hour periods[3], it is really difficult to manually analyze these data, which calls for automatic techniques to support the efficient diagnosis of heart diseases.

Machine learning is employed frequently as an artificial intelligence technique to facilitate automated analysis. Many machine learning techniques have been applied to medical-aided diagnosis, such as support vector machines[4], decision trees[5], random conditional fields[6], and recently developed deep learning methods[7]. However, most of these methods require large amounts of labeled data for training the model, which is an empirical problem that still needs to be solved. For example, large volumes of labeled ECG data are usually required as training samples for heart disease classification systems. Moreover, when machine learning approaches are applied to personalized medicine research, such as personalized heart disease research, the ECGs are often categorized based on the personal features of the patients, such as their gender and age. Thus, the problems caused by lacking of good ECG data are exacerbated before any subsequent analysis. Furthermore, maintaining the privacy of patients is always an issue that cannot be igored. However, the personal information and private clinical data obtained from patients are still likely to be illegally leaked. An optimal solution is to generate synthetic data without any private details to satisfy the requirements for research.

[1]School of Computer Science and Technology, Soochow University, Suzhou, 215006, China. [2]Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou, 215006, China. [3]School of Computer Science and Engineering, Changshu Institute of Technology, Changshu, 215500, China. [4]Institutes for Systems Genetics, West China Hospital, Sichuan University, Chengdu, 610041, China. Correspondence and requests for materials should be addressed to B.S. (email: Bairong.shen@scu.edu.cn)

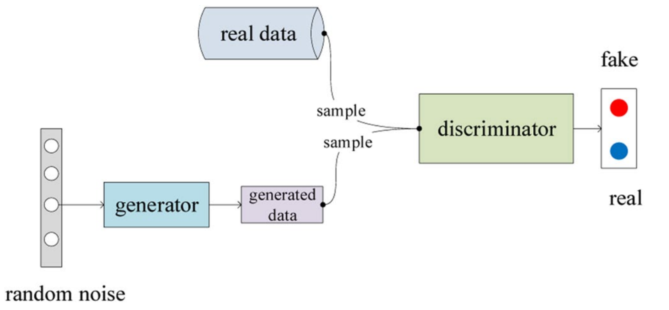

Figure 1. Architecture of the GAN.

Hence, it is very necessary to develop a suitable method for producing practical medical samples for disease research, such as heart disease. Several previous studies have investigated the generation of ECG data. McSharry *et al.* proposed a dynamic model based on three coupled ordinary differential equations[8], where real synthetic ECG signals can be generated by specifying heart rate or morphological parameters for the PQRST cycle. Clifford *et al.* used a nonlinear model to generate 24-hour ECG, blood pressure, and respiratory signals with realistic linear and nonlinear clinical characteristics[9]. Cao *et al.* designed an ECG system for generating conventional 12-lead signals[10]. However, most of these ECG generation methods are dependent on mathematical models to create artificial ECGs, and therefore they are not suitable for extracting patterns from existing ECG data obtained from patients in order to generate ECG data that match the distributions of real ECGs.

The generative adversarial network (GAN) proposed by Goodfellow in 2014 is a type of deep neural network that comprises a generator and a discriminator[11]. The generator produces data based on the noise data sampled from a Gaussian distribution, which is fitted to the real data distribution as accurately as possible. The inputs for the discriminator are real data and the results produced by the generator, where the aim is to determine whether the input data are real or fake. During the training process, the generator and the discriminator play a zero-sum game until they converge. GAN has been shown to be an efficient method for generating data, such as images.

In this study, we propose a novel model for automatically learning from existing data and then generating ECGs that follow the distribution of the existing data so the features of the existing data can be retained in the synthesized ECGs. Our model is based on a GAN architecture which is consisted of a generator and a discriminator. In the generator part, the inputs are noise data points sampled from a Gaussian distribution. We build up two layers of bidirectional long short-term memory (BiLSTM) networks[12], which has the advantage of selectively retaining the history information and current information. Moreover, to prevent over-fitting, we add a dropout layer. In the discriminator part, we classify the generated ECGs using an architecture based on a convolutional neural network (CNN). The discriminator includes two pairs of convolution-pooling layers as well as a fully connected layer, a softmax layer, and an output layer from which a binary value is determined based on the calculated one-hot vector. We used the MIT-BIH arrhythmia data set[13] for training. The results indicated that our model worked better than the other two methods, the deep recurrent neural network-autoencoder (RNN-AE)[14] and the RNN-variational autoencoder (RNN-VAE)[15].

Related Work

Generative Adversarial Network.

The GAN is a deep generative model that differs from other generative models such as autoencoder in terms of the methods employed for generating data and is mainly comprised of a generator and a discriminator. The generator produces data based on sampled noise data points that follow a Gaussian distribution and learns from the feedback given by the discriminator. The discriminator learns the probability distribution of the real data and gives a true-or-false value to judge whether the generated data are real ones. The two sub-models comprising the generator and discriminator reach a convergence state by playing a zero-sum game. Figure 1 illustrates the architecture of GAN.

The solution obtained by GAN can be viewed as a min-max optimization process. The objective function is:

$$\min_G \max_D V(D, G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{1}$$

where D is the discriminator and G is the generator. When the distribution of the real data is equivalent to the distribution of the generated data, the output of the discriminator can be regarded as the optimal result.

GAN has been successfully applied in several areas such as natural language processing[16,17], latent space learning[18], morphological studies[19], and image-to-image translation[20].

RNN.

Recurrent neural network has been widely used to solve tasks of processing time series data[21], speech recognition[22], and image generation[23]. Recently, it has also been applied to ECG signal denoising and ECG classification for detecting obstructions in sleep apnea[24]. RNN typically includes an input layer, a hidden layer, and an output layer, where the hidden state at a certain time t is determined by the input at the current time as well as by the hidden state at a previous time:

$$h_t = f(W_{ih}x_t + W_{hh}h_{t-1} + b_h), \tag{2}$$

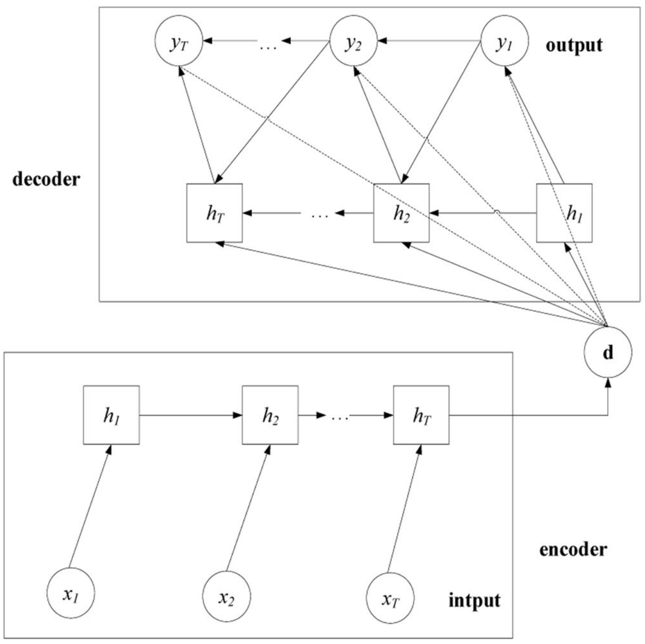

Figure 2. Illustration of the RNN-AE architecture.

$$o_t = g(W_{ho}h_t + b_o), \tag{3}$$

where f and g are the activation functions, x_t and o_t are the input and output at time t, respectively, h_t is the hidden state at time t, $W_{\{ih,hh,ho\}}$ represent the weight matrices that connect the input layer, hidden layer, and output layer, and $b_{\{h,o\}}$ denote the basis of the hidden layer and output layer.

RNN is highly suitable for short-term dependent problems but is ineffective in dealing with long-term dependent problems. The long short-term memory (LSTM)[25] and gated recurrent unit (GRU)[26] were introduced to overcome the shortcomings of RNN, including gradient expansion or gradient disappearance during training. The LSTM is a variation of an RNN and is suitable for processing and predicting important events with long intervals and delays in time series data by using an extra architecture called the memory cell to store previously captured information. LSTM has been applied to tasks based on time series data such as anomaly detection in ECG signals[27]. However, LSTM is not part of the generative models and no studies have employed LSTM to generate ECG data yet. The GRU is also a variation of an RNN, which combines the forget gate and input gate into an update gate to control the amount of information considered from previous time flows at the current time. The reset gate of the GRU is used to control how much information from previous times is ignored. GRUs have been applied in some areas in recent years, such as speech recognition[28].

RNN-AE and RNN-VAE. The autoencoder and variational autoencoder (VAE) are generative models proposed before GAN. Besides used for generating data[29], they were utilized to dimensionality reduction[30,31].

RNN-AE is an expansion of the autoencoder model where both the encoder and decoder employ RNNs. The encoder outputs a hidden latent code **d**, which is one of the input values for the decoder. In contrast to the encoder, the output and hidden state of the decoder at the current time depend on the output at the current time and the hidden state of the decoder at the previous time as well as on the latent code **d**. The goal of RNN-AE is to make the raw data and output for the decoder as similar as possible. Figure 2 illustrates the RNN-AE architecture[14].

VAE is a variant of autoencoder where the decoder no longer outputs a hidden vector, but instead yields two vectors comprising the mean vector and variance vector. A skill called the re-parameterization trick[32] is used to re-parameterize the random code **z** as a deterministic code, and the hidden latent code **d** is obtained by combining the mean vector and variance vector:

$$\mathbf{d} = \mu + \sigma \odot \varepsilon, \tag{4}$$

where μ is the mean vector, σ is the variance vector, and $\varepsilon \sim N(0, 1)$.

RNN-VAE is a variant of VAE where a single-layer RNN is used in both the encoder and decoder. This model is suitable for discrete tasks such as sequence-to-sequence learning and sentence generation.

Generation of Time Series Data. To the best of our knowledge, there is no reported study adopting the relevant techniques of deep learning to generate or synthesize ECG signals, but there are some related works on the generation of audio and classic music signals.

Methods for generating raw audio waveforms were principally based on the training autoregressive models, such as Wavenet[33] and SampleRNN[34], both of them using conditional probability models, which means that at

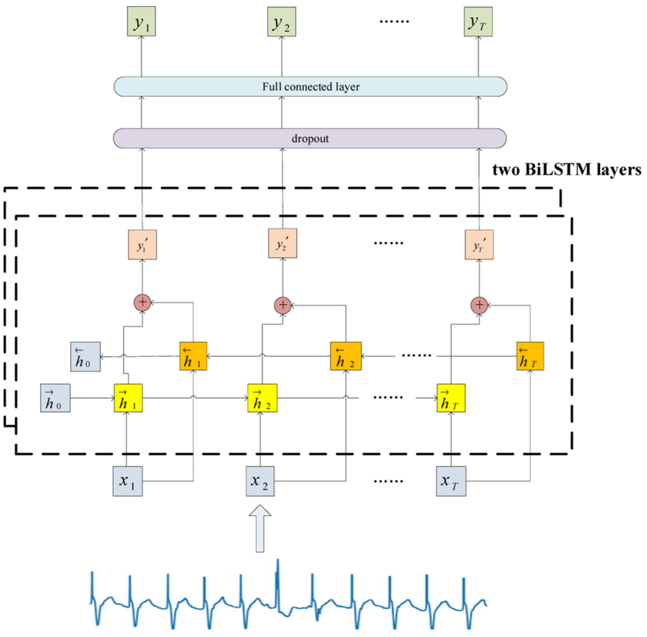

Figure 3. Architecture of the generator.

time *t* each sample is generated according to all samples at previous time steps. However, autoregressive settings tend to result in slow generation because the output audio samples have to be fed back into the model once each time, while GAN is able to avoid this disadvantage by constantly adversarial training to make the distribution of generated results and real data as approximate as possible.

Mogren *et al.* proposed a method called C-RNN-GAN[35] and applied it on a set of classic music. In their work, tones are represented as quadruplets of frequency, length, intensity and timing. Both the generator and the discriminator use a deep LSTM layer and a fully connected layer. Inspired by their work, in our research, each point sampled from ECG is denoted by a one-dimensional vector of the time-step and leads. Donahue *et al.* applied WaveGANs[36] from aspects of time and frequency to audio synthesis in an unsupervised background. WaveGAN uses a one-dimensional filter of length 25 and a great up-sampling factor. However, it is essential that these two operations have the same number of hyper parameters and numerical calculations. According to the above analysis, our architecture of GAN will adopt deep LSTM layers and CNNs to optimize generation of time series sequence.

Model Design

Overview of the Model.

We propose a GAN-based model for generating ECGs. Our model comprises a generator and a discriminator. The input to the generator comprises a series of sequences where each sequence is made of 3120 noise points. The output is a generated ECG sequence with a length that is also set to 3120. The input to the discriminator is the generated result and the real ECG data, and the output is $D(x) \in \{0, 1\}$. In the training process, G is initially fixed and we train D to maximize the probability of assigning the correct label to both the realistic points and generated points. We then train G to minimize $\log(1 - D(G(z)))$. The objective function is described by Eq. 5:

$$\min_{G_\theta} \max_{D_\phi} L_{\theta;\phi} = \frac{1}{N}\sum_{i=1}^{N}[\log D_\phi(x_i) + (\log(1 - D_\phi(G_\theta(z_i))))], \tag{5}$$

where N is the number of points, which is 3120 points for each sequence in our study, and θ and ϕ represent the set of parameters.

As CNN does not have recurrent connections like forgetting units as in LSTM or GRU, the training process of the models with CNN-based discriminator is often faster, especially in the case of long sequence data modeling. CNN has achieved excellent performance in sequence classification such as the text or voice sorting[37]. Many successful deep learning methods applied to ECG classification and feature extraction are based on CNN or its variants. Therefore, the CNN discriminator is nicely suitable to the ECG sequences data modeling.

Design of the Generator.

A series of noise data points that follow a Gaussian distribution are fed into the generator as a fixed length sequence. We assume that each noise point can be represented as a d-dimensional one-hot vector and the length of the sequence is T. Thus, the size of the input matrix is $T \times d$.

The generator comprises two BiLSTM layers, each having 100 cells. A dropout layer is combined with a fully connected layer. The architecture of the generator is shown in Fig. 3.

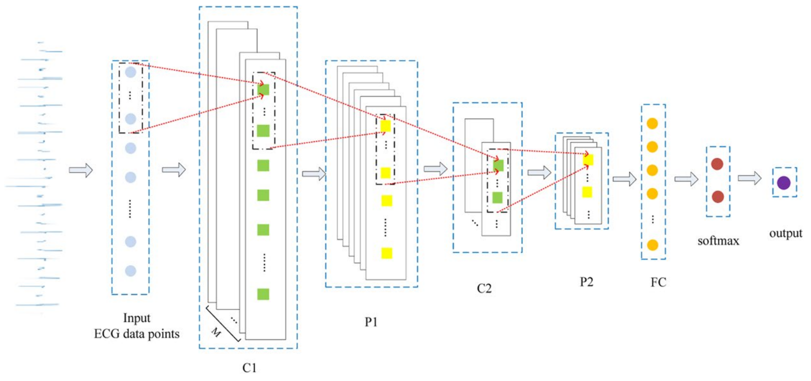

Figure 4. Structure of the CNN in the discriminator.

The current hidden state depends on two hidden states, one from forward LSTM and the other from backward LSTM. Eqs 6 and 7 are used to calculate the hidden states from two parallel directions and Eq. 9 calculates the output of the first BiLSTM layer at time t:

$$\overrightarrow{h_t^1} = \tanh\left(W^1_{i\overrightarrow{h}} x_t + W^1_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}^1_{t-1} + b^1_{\overrightarrow{h}}\right), \tag{6}$$

$$\overleftarrow{h_t^1} = \tanh\left(W^1_{i\overleftarrow{h}} x_t + W^1_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}^1_{t+1} + b^1_{\overleftarrow{h}}\right), \tag{7}$$

$$y_t^1 = \tanh\left(W^1_{\overrightarrow{h}o} \overrightarrow{h_t^1} + W^1_{\overleftarrow{h}o} \overleftarrow{h_t^1} + b_o^1\right), \tag{8}$$

where the output depends on $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, and h_0 is initialized as a zero vector.

Similarly, we obtain the output at time t from the second BiLSTM layer:

$$y_t = \tanh\left(W^2_{\overrightarrow{h}o} \overrightarrow{h_t^2} + W^2_{\overleftarrow{h}o} \overleftarrow{h_t^2} + b_o^2\right). \tag{9}$$

To prevent slow gradient descent due to parameter inflation in the generator, we add a dropout layer and set the probability to 0.5[38]. The output layer is a two-dimensional vector where the first element represents the time step and the second element denotes the lead.

Design of the Discriminator.

The architecture of discriminator is illustrated in Fig. 4. The pair of red dashed lines on the left denote a type of mapping indicating the position where a filter is moved, and those on the right show the value obtained by using the convolution operation or the pooling operation.

The sequence comprising ECG data points can be regarded as a time series sequence (a normal image requires both a vertical convolution and a horizontal convolution) rather than an image, so only one-dimensional (1-D) convolution need to be involved. We assume that an input sequence $x_1, x_2, \ldots x_T$ comprises T points, where each is represented by a d-dimensional vector. We set the size of filter to h*1, the size of the stride to k*1 ($k \ll h$), and the number of the filters to M. Therefore, the output size from the first convolutional layer is $M * [(T-h)/k+1] * 1$. The window for the filter is:

$$x_{l:r} = x_l \oplus x_{l+1} \oplus x_{l+2} \oplus \ldots \oplus x_r. \tag{10}$$

The values of l and r are determined by:

$$l = k * i + 1 - k \qquad l \in [1,\ T - h + 1], \tag{11}$$

$$r = k * i - k + h \qquad r \in [h,\ T], \tag{12}$$

where $1 \leq k*i+1 \leq T-h+1$ and $h \leq k*i-k+h \leq T$ $(i \in [1, (T-h)/k+1])$.

The returned convolutional sequence $c = [c_1, c_2, \ldots c_i, \ldots]$ with each c_i is calculated as

$$c_i = f(w * x_{l:r} + b), \tag{13}$$

where $w \in \mathbb{R}^{h \times d}$ a shared weight matrix, and f represents a nonlinear activation function.

The successor layer is the max pooling layer with a window size of a*1 and stride size of b*1. Each output from pooling p_j for the returned pooling result sequence $p = [p_1, p_2, \ldots p_j \ldots]$ is:

Layer	Feature Maps	Filter	Stride	Input Size	Output Size
Input	—	—	—	1*3120*1	—
C1	10	120*1	5*1	1*3120*1	10*601*1
P1	10	46*1	3*1	10*601*1	10*186*1
C2	5	36*1	3*1	10*186*1	5*51*1
P2	5	24*1	3*1	5*51*1	5*10*1
FC	5	—	—	5*10*1	25
Softmax	—	—	—	25	25
Output	—	—	—	25	1

Table 1. Parameters for each layer of the discriminator.

$$p_j = \max(c_{bj+1-b}, c_{bj+2-b}, \ldots c_{bj+a-b}). \tag{14}$$

After conducting double pairs of operations for convolution and pooling, we add a fully connected layer that connects to a softmax layer, where the output is a one-hot vector. The two elements in the vector represent the probability that the input is true or false. The function of the softmax layer is:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{2} e^{z_k}} (j = 1,\ 2). \tag{15}$$

In Table 1, C1 layer is a convolutional layer, with the size of each filter 120*1, the number of filters is 10 and the size of stride is 5*1. The output size of C1 is calculated by:

$$\frac{(W, H) - F + 2P}{S} + 1, \tag{16}$$

where (W, H) represents the input volume size (1*3120*1), F and S denote the size of kernel filters and length of stride respectively, and P is the amount of zero padding and it is set to 0. Thus, the output size of C1 is 10*601*1.

In Table 1, the P1 layer is a pooling layer where the size of each window is 46*1 and size of stride is 3*1. The output size of P1 is computed by:

$$\frac{(W, H) - F}{S} + 1, \tag{17}$$

where (W, H) represents the input volume size (10*601*1), F and S denote the size of each window and the length of stride respectively. Thus, calculated by Eq. 17, the output size of P1 is 10*186*1.

The computational principle of parameters of convolutional layer C2 and pooling layer P2 is the same as that of the previous layers. It needs to be emphasized that the amount of kernels filters of C2 is set to 5 factitiously. With pairs of convolution-pooling operations, we get the output size as 5*10*1. A fully connected layer which contains 25 neurons connects with P2. The last layer is the softmax-output layer, which outputs the judgement of the discriminator.

Experiments and Analyses

The Computing Platform. In the experiment, we used a computer with an Intel i7-7820X (8 cores) CUP, 16 GB primary memory, and a GeForce GTX 1080 Ti graphics processing unit (GPU). The operating system is Ubuntu 16.04LTS. We implemented the model by using Python 2.7, with the package of PyTorch and NumPy. Compared to the static platform, the established neural network in PyTorch is dynamic. The result of the experiment is then displayed by Visdom, which is a visual tool that supports PyTorch and NumPy.

Representation of ECG Data. We used the MIT-BIH arrhythmia data set provided by the Massachusetts Institute of Technology for studying arrhythmia in our experiments. We downloaded 48 individual records for training. Each record comprised three files, i.e., the header file, data file, and annotation file. Each data file contained about 30 minutes of ECG data. In each record, a single ECG data point comprised two types of lead values; in this work, we only selected one lead signal for training:

$$x_t = [x_t^{\alpha},\ x_t^{\beta}]^T, \tag{18}$$

$$v(x_t) = x_t^{\alpha}/200, \tag{19}$$

where x_t represents the ECG points at time step t sampled at 360 Hz, x_t^{α} is the first sampling signal value, and x_t^{β} is the second one. Both were divided by 200 to calculate the corresponding lead value. The number of ECG data points in each record was calculated by multiplying the sampling frequency (360 Hz) and duration of each record for about 650,000 ECG data points. Therefore, we used 31.2 million points in total.

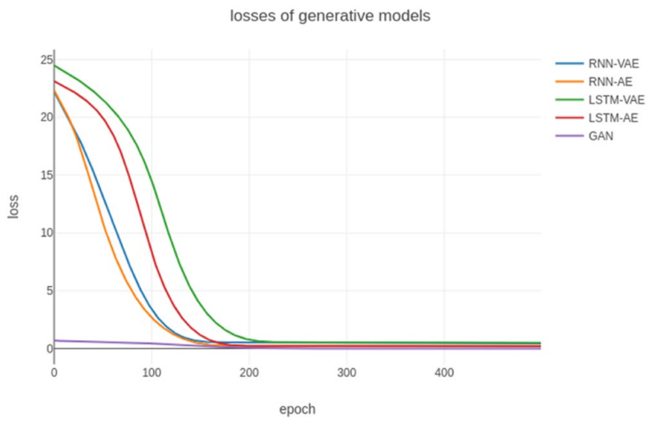

Figure 5. Losses of five generative models.

Training Results. First, we compared the GAN with RNN-AE and RNN-VAE. All of the models were trained for 500 epochs using a sequence of 3120 points, a mini-batch size of 100, and a learning rate of 10^{-5}. The loss of the GAN was calculated with Eq. 5 and the loss of RNN-AE was calculated as:

$$\max_{\theta} = \frac{1}{N}\sum_{i=1}^{N}\log p_{\theta}(y_i|x_i), \tag{20}$$

where θ is the set of parameters, N is the length of the ECG sequence, x_i is the ith point in the sequence, which is the input of for the encoder, and y_i is the ith point in the sequence, which is the output from the decoder.

The loss of RNN-VAE was calculated as:

$$\sum_{i=1}^{N}L(\theta,\ \phi:\ x_i) = \sum_{i=1}^{N} -KL(q_{\phi}(\vec{z}|x_i))\|p_{\theta}(\vec{z}) + E_{q_{\phi}(\vec{z}|x_i)}[\log p_{\theta}(x_i|\vec{z})], \tag{21}$$

where $p_{\theta}(\vec{z})$ is usually a standard prior $N \sim (0, 1)$, $q_{\phi}(\vec{z}|x)$ is the encoder, $p_{\theta}(x|\vec{z})$ is the decoder, and θ and ϕ are the sets of parameters for the decoder and encoder, respectively.

We extended the RNN-AE to LSTM-AE, RNN-VAE to LSTM-VAE, and then compared the changes in the loss values of our model with these four different generative models. Figure 5 shows the training results, where the loss of our GAN model was the minimum in the initial epoch, whereas all of the losses of the other models were more than 20. After 200 epochs of training, our GAN model converged to zero while other models only started to converge. At each stage, the value of the loss function of the GAN was always much smaller than the losses of the other models obviously.

We then compared the results obtained by the GAN models with those using a CNN, MLP (Multi-Layer Perceptron), LSTM, and GRU as discriminators, which we denoted as BiLSTM-CNN, BiLSTM-GRU, BiLSTM-LSTM, and BiLSTM-MLP, respectively. Each model was trained for 500 epochs with a batch size of 100, where the length of the sequence comprised a series of ECG 3120 points and the learning rate was 1×10^{-5}. Figure 6 shows the losses calculated of the four GAN discriminators using Eq. 5.

Figure 6 shows that the loss with the MLP discriminator was minimal in the initial epoch and largest after training for 200 epochs. The loss with the discriminator in our model was slightly larger than that with the MLP discriminator at the beginning, but it was obviously less than those of the LSTM and GRU discriminators. Eventually, the loss converged rapidly to zero with our model and it performed the best of the four models.

ECG Generation. Finally, we used the models obtained after training to generate ECGs by employing the GAN with the CNN, MLP, LSTM, and GRU as discriminators. The dim for the noise data points was set to 5 and the length of the generated ECGs was 400. Figure 7 shows the ECGs generated with different GANs.

Figure 7 shows that the ECGs generated by our proposed model were better in terms of their morphology. We found that regardless of the number of time steps, the ECG curves generated using the other three models were warped up at the beginning and end stages, whereas the ECGs generated with our proposed model were not affected by this problem.

We then evaluated the ECGs generated by four trained models according to three criteria. The distortion quantifies the difference between the original signal and the reconstructed signal. We evaluated the difference between the real data and the generated points with the percent root mean square difference (PRD)[39], which is the most widely used distortion measurement method.

The generated points were first normalized by:

$$x_{[n]} = \frac{x_{[n]} - x_{\max}}{x_{\max} - x_{\min}}. \tag{22}$$

The PRD was calculated as:

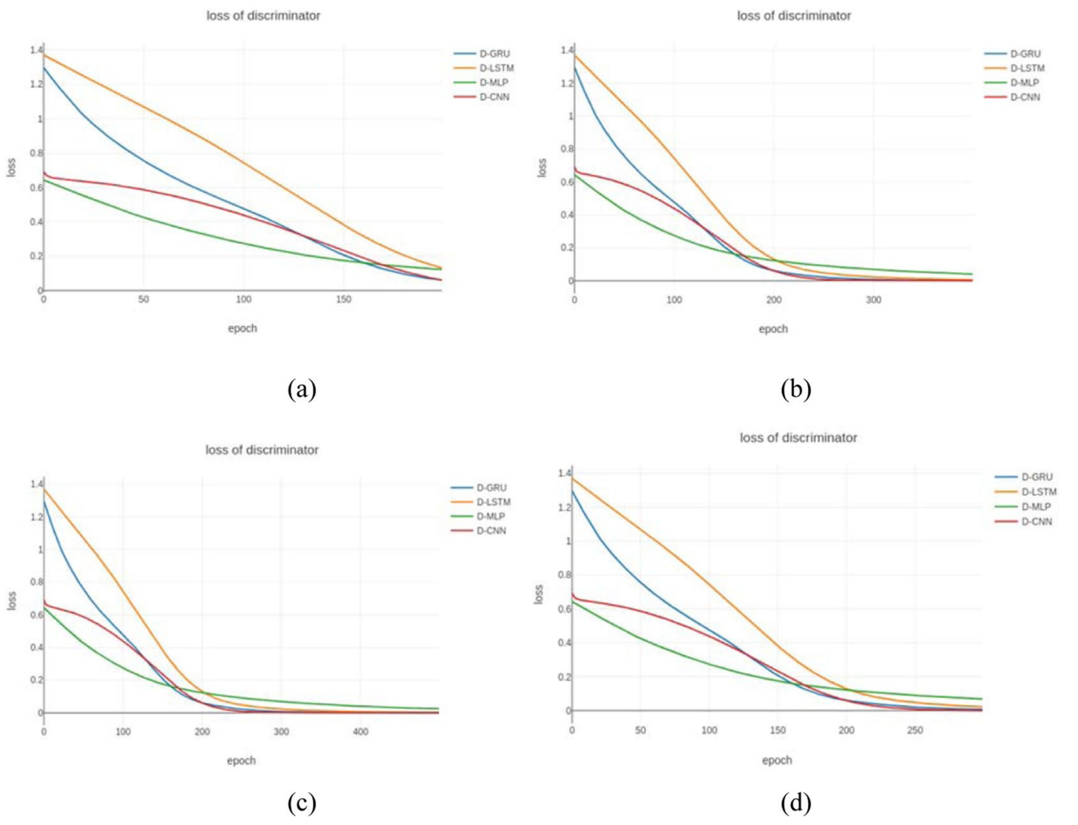

Figure 6. Loss of each type of discriminator. The four lines represent the discriminators based mainly on the structure with the CNN (red line), MLP (green line), LSTM (orange line), and GRU (blue line). (**a–d**) Represent the results after 200, 300, 400, and 500 epochs of training.

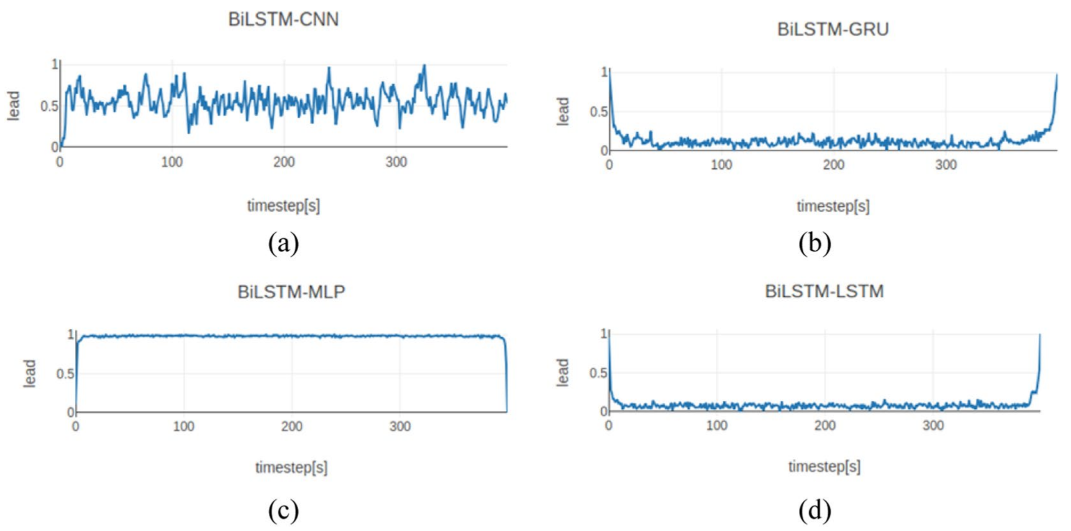

Figure 7. Results generated using different discriminator structures. (**a–d**) Represent the results obtained when the discriminator used the CNN, GRU, MLP, and LSTM respectively.

$$PRD = \sqrt{\frac{\sum_{n=1}^{N}(x_{[n]} - \widehat{x_{[n]}})^2}{\sum_{n=1}^{N}(x_{[n]})^2} \times 100}, \tag{23}$$

where $x_{[n]}$ is the nth real point, $\widehat{x_{[n]}}$ is the nth generated point, and N is the length of the generated sequence.

Method	PRD	RMSE	FD
BILSTM-CNN GAN	**66.408**	**0.276**	**0.756**
RNN-AE GAN	121.877	0.506	0.969
LSTM-AE GAN	148.650	0.618	0.996
RNN-VAE GAN	146.566	0.609	0.982
LSTM-VAE GAN	145.978	0.607	0.975

Table 2. Results of evaluate metrics for different generative models.

Method	PRD	RMSE	FD
BiLSTM-CNN GAN	**51.799**	**0.215**	**0.803**
BiLSTM-GRU	74.047	0.308	0.853
BiLSTM-LSTM	84.795	0.352	0.901
BiLSTM-MLP	147.732	0.614	0.989

Table 3. Results of evaluate metrics for GANs with different discriminators.

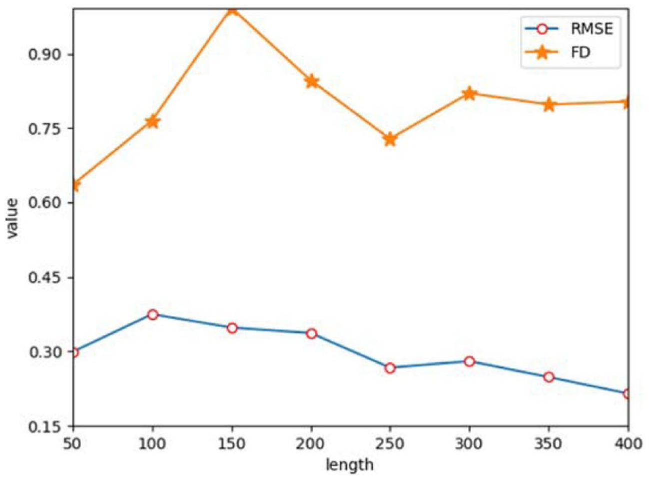

Figure 8. Results of RMSE and FD by different specified lengths.

The root mean square error (RMSE)[39] reflects the stability between the original data and generated data, and it was calculated as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_{[n]} - \widehat{x_{[n]}})^2}. \tag{24}$$

The Fréchet distance (FD)[40] is a measure of similarity between curves that takes into consideration the location and ordering of points along the curves, especially in the case of time series data. A lower FD usually stands for higher quality and diversity of generated results.

Let P be the order of points along a segment of realistic ECG curve, and Q be the order of points along a segment of a generated ECG curve: $\sigma(P) = (u_1,\ u_2,\ \dots u_p)$, $\sigma(Q) = (\nu_1,\ \nu_2,\ \dots \nu_q)$. Then we can get a sequence which consists of couple of points: $\{(u_{a_1}, v_{b_1}),\ \dots(u_{a_m}, v_{b_m})\}$. The length $||d||$ of this sequence is computed by:

$$||d|| = \max_{i=1,\dots m} d(u_{a_i}, v_{b_i}), \tag{25}$$

where d represents the Euclidean distance. Essentially, we have $a_{i+1} = a_i$ or $a_{i+1} = a_i + 1$ and $b_{i+1} = b_i$ as prerequisites.

Finally, the discrete Fréchet distance is calculated as:

$$FD(P, Q) = \min\{||d||\} \tag{26}$$

Table 2 shows that our model has the smallest metric values about PRD, RMSE and FD compared with other generative models.

We can see that the FD metric values of other four generative models fluctuate around 0.950. The RMSE and PRD of these models are much smaller than that of the BiLSTM-CNN GAN. This indicates that except for RNN-AE, the corresponding PRD and RMSE of LSTM-AE, RNN-VAE, LSTM-VAE are fluctuating between 145.000 to 149.000, 0.600 to 0.620 respectively because of their similar architectures. Based on the results shown in Table 2, we can conclude that our model is the best in generating ECGs compared with different variants of the autocoder. Table 3 shows that our proposed model performed the best in terms of the RMSE, PRD and FD assessment compared with different GANs.

Table 3 demonstrated that the ECGs obtained using our model were very similar to the standard ECGs in terms of their morphology. In addition, the LSTM and GRU are both variations of RNN, so their RMSE and PRD values were very similar.

From the results listed in Tables 2 and 3, we can see that both of RMSE and FD values are between 0 and 1. Under the BiLSTM-CNN GAN, we separately set the length of the generated sequences and obtain the corresponding evaluation values. It is well known that under normal circumstances, the average heart rate is 60 to 100 in a second. Therefore, the normal cardiac cycle time is between 0.6 s to 1 s. Based on the sampling rate of the MIT-BIH, the calculated length of a generated ECG cycle is between 210 and 360. Figure 8 shows the results of RMSE and FD by different specified lengths from 50–400. From Fig. 8, we can conclude that the quality of generation is optimal when the generated length is 250 (RMSE: **0.257**, FD: **0.728**).

Conclusion

To address the lack of effective ECG data for heart disease research, we developed a novel deep learning model that can generate ECGs from clinical data without losing the features of the existing data. Our model is based on the GAN, where the BiLSTM is used as the generator and the CNN is used as the discriminator. After training with ECGs, our model can create synthetic ECGs that match the data distributions in the original ECG data. Our model performed better than other two deep learning models in both the training and evaluation stages, and it was advantageous compared with other three generative models at producing ECGs. The ECGs synthesized using our model were morphologically similar to the real ECGs.

References

1. Wang, H. *et al*. Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: a systematic analysis for the Global Burden of Disease Study 2015. *The Lancet* **388**(10053), 1459–1544, https://doi.org/10.1016/S0140-6736(16)31012-1 (2016).
2. Lilly, L. S. Pathophysiology of heart disease: a collaborative project of medical students and faculty. *Lippincott Williams & Wilkins*, (2015).
3. George, S. *et al*. Computerized extraction of electrocardiograms from continuous 12- lead holter recordings reduces measurement variability in a thorough QT study. *The Journal of Clinical Pharmacology* **52**(12), 1891–1900, https://doi.org/10.1177/0091270011430505 (2012).
4. Kampouraki, A., Manis, G. & Nikou, C. Heartbeat time series classification with support vector machines. *IEEE Transactions on Information Technology in Biomedicine* **13**(4), 512–518, https://doi.org/10.1109/TITB.2008.2003323 (2009).
5. Zhang, L., Peng, H. & Yu, C. An approach for ECG classification based on wavelet feature extraction and decision tree. In *International Conference on Wireless Communications and Signal Processing (WCSP)*, 1–4, https://doi.org/10.1109/WCSP.2010.5633782 (2010).
6. Wei, Q. *et al*. Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks. *Database* **10**, 1–8, https://doi.org/10.1093/database/baw140 (2016).
7. Benali, R., Reguig, F. B. & Slimane, Z. H. Automatic classification of heartbeats using wavelet neural network. *Journal of medical systems* **36**, 883–892, https://doi.org/10.1007/s10916-010-9551-7 (2012).
8. McSharry, P. E. *et al*. A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomedical Engineering* **50**, 289–294, https://doi.org/10.1109/TBME.2003.808805 (2003).
9. Clifford, G. & McSharry, P. Generating 24-hour ECG, BP and respiratory signals with realistic linear and nonlinear clinical characteristics using a nonlinear model. *Computers in Cardiology*, 709–712, https://doi.org/10.1109/CIC.2004.1443037 (2004).
10. Cao, H. *et al*. Design and evaluation of a novel wireless three-pad ECG system for generating conventional 12-lead signals. *the Fifth International Conference on Body Area Networks*, 84–90, https://doi.org/10.1145/2221924.2221942 (2010).
11. Goodfellow, I. J. *et al*. Generative adversarial networks. *Advances in Neural Information Processing Systems* **3**, 2672–2680, https://arxiv.org/abs/1406.2661 (2014).
12. Yao, Y. & Huang, Z. Bi-directional LSTM recurrent neural network for Chinese word segmentation. *International Conference on Neural Information Processing*, 345–353, https://arxiv.org/abs/1602.04874 (2016).
13. Torres-Alegre, S. *et al*. Artificial Metaplasticity: Application to MITBIH Arrhythmias Database. Artificial Computation in Biology and Medicine, Springer International Publishing (2015).
14. Cho, K. *et al*. Learning phrase representations using RNN encoder--decoder for statistical machine translation. *Empirical Methods in Natural Language Processing*, 1724–1734, https://arxiv.org/abs/1406.1078 (2014).
15. Kingma, D. P. & Welling, M. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 1–14, https://arxiv.org/abs/1312.6114 (2014).
16. Press, O. *et al*. Language generation with recurrent generative adversarial networks without pre-training. *the 1st Workshop on Learning to Generate Natural Language at ICML 2017*, 1–5, https://arxiv.org/abs/1706.01399 (2017).
17. Li, J. *et al*. Adversarial learning for neural dialogue generation. *Empirical Methods in Natural Language Processing*, 2157–2169, https://arxiv.org/abs/1701.06547 (2017).
18. Chen, X. *et al*. InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. *Advances in Neural Information Processing Systems*, 2180–2188, https://arxiv.org/abs/1606.03657 (2016).
19. Wang, Z. *et al*. Defo-Net: Learning body deformation using generative adversarial networks. *International Conference on Robotics and Automation*, https://arxiv.org/abs/1804.05928, 2440–2447 (2018).
20. Zhu J. *et al*. Unpaired image-to-image translation using cycle-consistent adversarial networks. *International Conference on Computer Vision*, 2242–2251, https://doi.org/10.1109/iccv.2017.244 (2017).
21. Hüsken, M. & Stagge, P. Recurrent neural networks for time series classification. *Neurocomputing* **50**, 223–235, https://doi.org/10.1016/S0925-2312(01)00706-8 (2003).
22. Graves, A. *et al*. Speech recognition with deep recurrent neural networks. *International Conference on Acoustics, Speech, and Signal Processing*, 6645–6649, https://doi.org/10.1109/ICASSP.2013.6638947 (2013).
23. Gregor, K. *et al*. Draw: A recurrent neural network for image generation. *International Conference on Machine Learning*, 1462–1471, https://arxiv.org/abs/1502.04623 (2015).

24. Cheng, M. *et al.* Recurrent neural network based classification of ecg signal features for obstruction of sleep apnea detection. *IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC)*, 199–202, https://doi.org/10.1109/CSEEUC.2017.220 (2017).
25. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation* **9**, 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
26. Chung, J. *et al.* Gated feedback recurrent neural networks. *International Conference on Machine Learning*, 2067–2075, https://arxiv.org/abs/1502.02367 (2015).
27. Chauhan, S. & Vig, L. Anomaly detection in ECG time signals via deep long short-term memory networks. *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 1–7, https://doi.org/10.1109/DSAA.2015.7344872 (2015).
28. Ravanelli, M. *et al.* Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence* **2**, 92–102, https://doi.org/10.1109/tetci.2017.2762739 (2018).
29. Bowman, S. R. *et al.* Generating sentences from a continuous space. *Conference on Computational Natural Language Learning*, 10–21, https://doi.org/10.18653/v1/K16-1002 (2016).
30. Wang, J., He, H. & Prokhorov, D. V. A folded neural network autoencoder for dimensionality reduction. *Procedia Computer Science* **13**, 120–127, https://doi.org/10.1016/j.procs.2012.09.120 (2012).
31. Zabalza, J. *et al.* Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing* **185**, 1–10, https://doi.org/10.1016/j.neucom.2015.11.044 (2016).
32. Kingma, D. P. *et al.* Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, 2575–2583, https://arxiv.org/abs/1506.02557 (2015).
33. Den, Oord A. V. *et al.* Wavenet: a generative model for raw audio. *the 9th ISCA Speech Synthesis Workshop*, 1–15, https://arxiv.org/abs/1609.03499 (2016).
34. Mehri, S. *et al.* SampleRNN: an unconditional rnd-to-rnd neural audio generation model. *International Conference on Learning Representations*, 1–11, https://arxiv.org/abs/1612.07837 (2017).
35. Mogren, O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *Advances in Neural Information Processing systems*, 1–6, https://arxiv.org/abs/1611.09904 (2016).
36. Donahue, C., McAuley, J. & Puckette, M. Synthesizing audio with GANs. *the 6th International Conference on Learning Representations*, 1–6, (2018).
37. Kim, Y. Convolutional neural networks for sentence classification. *Empirical Methods in Natural Language Processing*, 1746–1751, https://doi.org/10.3115/v1/D14-1181 (2014).
38. Gal, Y. & Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. *Advances in Neural Information Processing Systems*, 1027–1035, https://arxiv.org/abs/1512.05287 (2016).
39. Almahamdy, M. & Riley, H. B. Performance study of different denoising methods for ECG signals. *Procedia Computer Science* **37**(37), 325–332, https://doi.org/10.1016/j.procs.2014.08.048 (2014).
40. Aronov B. *et al.* Fréchet distance for curves, revisited. *European Symposium on Algorithms*, 52–63, https://doi.org/10.1007/11841036_8 (2006).

Acknowledgements
This study was supported by the National Natural Science Foundation of China (61303108, 61373094, and 61772355), Jiangsu College Natural Science Research Key Program (17KJA520004), Suzhou Key Industries Technological Innovation-Prospective Applied Research Project (SYG201804), and Program of the Provincial Key Laboratory for Computer Information Processing Technology (Soochow University) (KJS1524).

Author Contributions
F.Z. and F.Y. performed the computational analyses; F.Z. and Y.F. performed the validation work; F.Z., F.Y. and Q.L. wrote the manuscript; B.S. coordinated the study.

Additional Information
Competing Interests: The authors declare no competing interests.