Expert Systems With Applications 135 (2019) 237–248

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Probabilistic grammar-based neuroevolution for physiological signal classification of ventricular tachycardia

Pak-Kan Wong[a,*], Kwong-Sak Leung[a], Man-Leung Wong[b]

[a] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Sha Tin, Hong Kong
[b] Department of Computing and Decision Sciences, Lingnan University, Tuen Mun, Hong Kong

ARTICLE INFO

Article history:
Received 25 July 2018
Revised 16 May 2019
Accepted 5 June 2019
Available online 6 June 2019

Keywords:
Physiological signal classification
Heart disease
Neuroevolution
Probabilistic grammar
Genetic programming
Deep neural network

ABSTRACT

Ventricular tachycardia is a rapid heart rhythm that begins in the lower chambers of the heart. When it happens continuously, this may result in life-threatening cardiac arrest. In this paper, we apply deep learning techniques to tackle the problem of the physiological signal classification of ventricular tachycardia, since deep learning techniques can attain outstanding performance in many medical applications. Nevertheless, human engineers are required to manually design deep neural networks to handle different tasks. This can be challenging because of many possible deep neural network structures. Therefore, a method, called ADAG-DNE, is presented to automatically design deep neural network structures using deep neuroevolution. Our approach defines a set of structures using probabilistic grammar and searches for best network structures using Probabilistic Model Building Genetic Programming. ADAG-DNE takes advantages of the probabilistic dependencies found among the structures of networks. When applying ADAG-DNE to the classification problem, our discovered model achieves better accuracy than AlexNet, ResNet, and seven non-neural network classifiers. It also uses about 2% of parameters of AlexNet, which means the inference can be made quickly. To summarize, our method evolves a deep neural network, which can be implemented in expert systems. The deep neural network achieves high accuracy. Moreover, it is simpler than existing deep neural networks. Thus, computational efficiency and diagnosis accuracy of the expert system can be improved.

1. Introduction

Cardiovascular disease is the leading cause of death among people. According to the report released by the American Heart Association (Benjamin et al., 2018), cardiac arrest as an underlying cause of death in 2015 was 17668; any-mention mortality in 2015 was 366807. If patients suffer from endstage renal disease (a type of kidney disease), arrhythmias (i.e. heart rhythm disorder) and sudden cardiac death accounts for nearly 40% of the deaths (Benjamin et al., 2018). In this paper, we are focusing on the diagnosis of ventricular tachycardia in Intensive Care Unit (ICU). Ventricular tachycardia is a heart arrhythmia initiated by abnormal electrical signals in the lower chambers of the heart (Berbari, Scherlag, Hope, & Lazzara, 1978; Bradfield, Boyle, & Shivkumar, 2017; Uther, Dennett, Duffy, Freedman, & Tan, 1979). An ICU in a hospital is a facility dedicated to providing life support and monitoring in patients who are critically ill, for instance, life-threatening illness, injuries, and multiple organ failures. ICU care is important to other medical services, including surgery (World Health Organization, 2003), or care for patients with complications of diseases (Baker, 2009). Higher quality of ICU will also increase citizen confidence in the health care system (Riviello, Letchford, Achieng, & Newton, 2011). In clinical management of ICU, real-time physiological measurement systems help clinicians to continually monitor the physiological status of patients. For example, pulse oximeter provides the oxygen saturation values and shows the plethysmographic waveform of the pulse signal over time (Shamir, Eidelman, Floman, Kaplan, & Pizov, 1999). A report on 2016 Get With The Guidelines programs shows that the location of adult in-hospital cardiac arrest was 53.7% in the ICU, operating room, or emergency department (Benjamin et al., 2018). Among 16.1% of these incidents, the initial recorded cardiac rhythm was ventricular fibrillation, ventricular tachycardia, or shockable (Benjamin et al., 2018). Therefore, there is a monitoring system in an ICU to detect if ventricular tachycardia happens. When ventricular tachycardia happens continuously, this may result in life-threatening cardiac arrest. When a life-threatening situation is detected, an alarm will be raised. Sometimes, an alarm may not be clinically

* Corresponding author.
E-mail addresses: pkwong@cse.cuhk.edu.hk (P.-K. Wong), ksleung@cse.cuhk.edu.hk (K.-S. Leung), mlwong@ln.edu.hk (M.-L. Wong).

https://doi.org/10.1016/j.eswa.2019.06.012
0957-4174/

significant. We call this a false alarm. False alarms in the ICU decreases the quality of care due to the noise disruptions and desensitization to warnings (Donchin & Seagull, 2002). This can lengthen the stay for recovery and increase the load on the health care system (Chambrin, 2001). It is also reported that less than 10% of alarms are associated with therapeutic modification (Chambrin et al., 1999; Lawless, 1994). Can the accuracy of the system be improved?

Recently, Deep Neural Network (DNN) is a powerful machine learning technique and attains outstanding performance in many applications. It has been applied in many medical applications, such as annotation of mitosis in breast cancer histology images (Cireşan, Giusti, Gambardella, & Schmidhuber, 2013), identification of skin lesions (Esteva et al., 2017), and detection of standard scan plane during fetal abnormality screening (Baumgartner et al., 2016). There has been much interest in applying DNN in other medical applications. Even a slight improvement in the performance implies an early diagnosis of diseases which can be critical to the treatment of patients. We believe that the physiological status of patients can also be diagnosed using DNN.

However, manually configuring of a DNN is complicated. Because the large search space, which is called the design space, of DNN covers many aspects, such as the topology of the network and the learning parameters. Properly designing DNN can be complicated to human engineers, medical experts, or other people who are knowledgeable in other application areas. Secondly, an expert may have data collected from a group of heterogeneous sensors. Understanding the nature of the signals from these sensors and performing feature engineering can be time-consuming. In the early stage of the study, it may be better to use automatic learning approaches on these signals and prioritize the signal analysis afterwards. This can increase the productivity of human engineers and medical experts. In this paper, we explore a physiological application of Deep Neuroevolution (DNE) to the automatically design of Convolutional Neural Network (CNN), which has not been done in the past.

The main contribution of this paper is that we propose a new procedure to evolve DNN structures from a set of convolutional layers, which are called modules, using Probabilistic Model Building Genetic Programming (PMBGP) approach. Theoretically, it is demonstrated that simpler and better DNNs can be evolved by evolutionary computation. Moreover, we also showed that structural dependencies within components in DNN can be encoded in grammar and then used by DNE. This enhances the search space modeling of DNNs during DNE. Finally, it is demonstrated that DNE is applicable to, not only image classification, but also physiological signal classification. A set of DNN structures are specified by a set of rules in Probabilistic Context-Sensitive Grammar (PCSG) in PMBGP. Each rule tells how a network structure is formed, how modules are connected, and which combinations of modules are preferred. In each iteration, the rules are updated iteratively based on the feedback from the fitness evaluation to guide evolution. There are several advantages to this approach. First of all, it explicitly models the preference for combinations among different modules via probabilistic dependencies in the set of DNNs. The probabilistic dependencies are captured by a set of Bayesian networks associated with every rule. The task to train a DNN is computationally expensive. As PMBGP approach can automatically learn the rules to compose the network modules, it can reduce the number of times of training DNNs with inappropriate structures. Secondly, since the DNN is represented in grammar, it is now possible to discover new forms of regularities and extract new traits to better optimize the DNN structures. The system can learn and decide what components to be inserted using context, i.e. information about the location of a new component concerning other existing components in a network structure. The context information can be nicely incorporated in PCSG while this is also the first study to apply PCSG and PMBGP on DNE. Experts in DNN can study the patterns and learn from the evolved design.

2. Related works

2.1. Neuroevolution

The development of neuroevolution began in the late 1980s (Montana & Davis, 1989). There are three aspects of Artificial Neural Network (ANN) and DNN which can be evolved: 1) evolution of connection weights; 2) evolution of architectures; 3) evolution of learning algorithms. The evolution of these configurations has been proposed in neuroevolution of ANN and reviewed in Yao (1999). These configurations can be controlled through hyperparameters and meta-heuristics. Evolution of connection weights is about finding a set of real values for ANN. Genetic Algorithm (GA) can encode the weights in a vector (Holland, 1992). Covariance Matrix Adaptation Evolution Strategy (CMA-ES) has been applied to the adaptation of weights on fixed ANN structure (Igel, 2003). CMA-ES is a stochastic, derivative-free approach for numerical optimization (Hansen & Ostermeier, 2001). Gomez and Miikkulainen (1997) proposed to evolve a hidden layer in ANN with recurrent connections to tackle a prey capture problem. Liang and Miikkulainen (2015) applied neuroevolution to control helicopter hovering.

The evolution of architectures is about designing the connections and choosing the transfer function of each node in ANN. One way is to encode the connections in a connection matrix using a binary string (Dasgupta & McGregor, 1992). It is also possible to represent the connections in an adjacency list. A node is transformed by dividing into more nodes. Liu et al. (2017a) evolved CNNs using Sequential Model-Based Optimization (SMBO) by gradually increasing the complexity of the network. In particular, they apply surrogate models to estimate the performance of the structures to improve the search efficiency. AmoebaNets introduces a new tournament selection scheme, which replaces the oldest individual by an offspring from the best of a group of random sample of individuals when evolving DNN (Real, Aggarwal, Huang, & Le, 2018). Rawal and Miikkulainen (2018) evolved a recurrent node using Genetic Programming (GP) and applied Long-Short Term Memory (LSTM) as a surrogate model to estimate the performance of the network. The authors tested their approach on creating homogeneous and heterogeneous Recurrent Neural Network (RNN) layers. Cartesian Genetic Programming (CGP) (Miller & Thomson, 2000) can be adopted to optimize CNN architectures (Suganuma, Shirakawa, & Nagao, 2017).

In addition, the evolution of learning algorithms is about how to adjust the connection weights during learning from data. This includes the rules to update the weights and choices of learning algorithms. In the context of multi-task learning, evolution can be applied to create a learning plan for the tasks.

Co-evolution of the above configurations is also viable. NeuroEvolution of Augmenting Topologies (NEAT) introduces historical markings, applies niching, incrementally grows structure, and evolves weights to achieve significant performance gains. Compositional Pattern Producing Network (CPPN) encodes the structural relationships in the complicated developmental process (i.e. growth from a small starting point to a mature form) without simulating itself (Stanley, 2007). In the same paper, the authors propose CPPN NEAT to evolve the composition of transfer functions and produce patterns of repetition with variation. CoDeepNEAT evolves blueprints of network topology and 15 hyperparameters. A blueprint contains some placeholders for DNN layers, such as convolutional, feed-forward and recurrent layers.

The hyperparameters specify the type of a layer of DNN and the settings in the gradient-based learning algorithm.

2.2. Grammar-based neuroevolution

Grammar is an alternative representation of the configurations in ANN and DNN. Grammar-Based Genetic Programming (GBGP) is designed to search within the possible configurations defined by the grammar (Whigham, 1995; Wong & Leung, 1995). For example, Grammatical Evolution (GE) (O'Neill & Ryan, 2003) evolve programs from Backus-Naur Form grammar, which is a Context-Free Grammar (CFG). Prior works evolve the network structure of a hidden layer in feed-forward neural networks using GE while optimizes the weights using back-propagation (Soltanian, Tab, Zar, & Tsoulos, 2013) or GA (Ahmadizar, Soltanian, AkhlaghianTab, & Tsoulos, 2015). Tsoulos, Gavrilis, and Glavas (2008) applied GE to construct the structure and set the weights of a hidden layer in feed-forward neural networks. Jung and Reggia (2006) presented a high-level descriptive language to represent modules and inter-layer connections hierarchically so as to enhance human readability and understandability of the search space. The language can represent Elman network, which is a three-layer recurrent network (Elman, 1990). Their approach can evolve the connectivity, hyperparameters, and learning rules. Cellular Encoding grows a graph from a node, while optimizes the structure and weights of boolean ANN at the same time (Gruau & Whitley, 1993). Fuzzy Petri nets can be evolved using Cellular Encoding (Wong, 1998).

Loshchilov and Hutter (2016) applied CMA-ES to evolve 19 hyperparameters, such as selection pressure, batch size, dropout rate, number of filters, and learning rate of the optimizer to perform classification on the MNIST dataset using deep CNN. Their network fixes the number of layers. Baldominos, Saez, and Isasi (2017) and Assunção, Lourenço, Machado, and Ribeiro (2018) aimed to cover the evolution of all aspects of design in deep CNN, including the structures, the activation functions, and the learning hyperparameters. The configurations are encoded in chromosome and grammar so that they can be searched via GA and GE respectively (Baldominos et al., 2017). They were able to design deep CNN for human activity recognition in sensor-rich environments (Baldominos, Saez, & Isasi, 2018).

2.3. Probabilistic model building genetic programming

Mühlenbein and Paass (1996) introduces Probabilistic Model Building Genetic Algorithm (PMBGA) or Estimation of Distribution Algorithm (EDA). Given an optimization problem, the objective of PMBGA is to gradually learn a probabilistic model which can (frequently) generate the best solution(s). Comprehensive surveys of these algorithms can be found in studies by Larrañaga and Lozano (2001) and Hauschild and Pelikan (2011). Instead of representing each solution in an array of values, PMBGP approaches represent each solution in a tree structure. For example, probabilistic prototype tree model-based methods operate on a fixed-length chromosome ((Hasegawa & Iba, 2008; Salustowicz & Schmidhuber, 1997; Sato, Hasegawa, Bollegala, & Iba, 2012)). In probabilistic grammar model-based methods, a grammar is incorporated to provide an alternative representation of the trees. There is such a wide variety of grammars to choose from, such as Stochastic CFG (Ratle & Sebag, 2001; Shan, McKay, Abbass, & Essam, 2003), Probabilistic Context-Free Grammar (PCFG) with Latent Annotations (Hasegawa, 2012; Hasegawa & Iba, 2009), Ant Tree Adjoining Grammars (Abbass, Hoai, & McKay, 2002), PCSG (Wong, Lo, Wong, & Leung, 2014a; 2014b), and hierarchical PCSG (Wong, Wong, & Leung, 2016).

2.4. Classification tasks on ECG signals

For the classification tasks on Electrocardiogram (ECG) signals, there are varieties of methods, such as harmonic analysis (Dzwonczyk, Brown, & Werman, 1990; Tripathy et al., 2018), time-frequency analysis (Millet-Roig, Rieta-Ibanez, Vilanova, Mocholi, & Chorro, 1999), wavelet-based analysis (Balasundaram, Masse, Nair, & Umapathy, 2013; Namarvar & Shahidi, 2004), high-order statistics (Martis et al., 2013), complexity measures (Acharya et al., 2016), independent component analysis (Sarfraz, Khan, & Li, 2014), and so on. Individually considering the signal features, such as magnitude features and phase features obtained from harmonic analysis, during classification often cannot achieve similar performance across multiple data sets due to the preselection of signals (Amann, Tratnig, & Unterkofler, 2005). Therefore, it is necessary to combine different signal features to make better prediction. Machine learning techniques, such as Support Vector Machine (SVM) (Alonso-Atienza, Morgado, Fernandez-Martinez, García-Alberola, & Rojo-Alvarez, 2014; Polat, Akdemir, & Güneş, 2008), decision tree (Xu, Wang, Zhang, Ping, & Feng, 2018), k-nearest neighbour classi er (KNN) (Arif, Malagore, & Afsar, 2012), and random forest (Masetic & Subasi, 2016), have been proposed to combine signal features. As an alternative, DNN can perform representation learning. For example, CNN learns different feature extractors and performs feature selection automatically. For the detection of ventricular ectopic beats and supraventricular ectopic beats, Kiranyaz, Ince, and Gabbouj (2016) developed a 1-D CNN for real-time patient-specific ECG classification. Zubair, Kim, and Yoon (2016) classified every beat in the signals into five categories using CNN. For the classification of shockable and non-shockable life-threatening ventricular arrhythmias from ECG signals, a 11-layer CNN was recently proposed by Acharya et al. (2018). Yıldırım, Pławiak, Tan, and Acharya (2018) utilized a 16-layer CNN of cardiac arrhythmia detection involving 17 classes. Evolutionary algorithms can be combined with CNN. Recently, an evolutionary-neural system was developed for the automatic recognition of myocardium dysfunctions by Pławiak (2018). This system utilized GA to optimize signal feature selection, and classifier parameters of probabilistic neural network (Specht, 1990) and radial basis function neural network (Broomhead & Lowe, 1988).

Our Adaptive Grammar-based Deep Neuroevolution (ADAG-DNE) method is a grammar-based neuroevolution method and uses Grammar-based Genetic Programming with Bayesian Network (BGBGP) to model probabilistic dependencies among the network modules (Wong et al., 2014a). In the following sections, the ADAG-DNE system is described. We begin by explaining the search space.

3. Deep neural network structure search space

A DNN model contains a DNN structure of several DNN modules and a set of weights. A DNN topology is a specification describing a set of DNN structures of interest. This specification is described using a grammar and a translation program. A translation program converts a parse tree, which can also be called an individual, derived from the grammar to a DNN structure.

In this paper, structures of DNN are built from eight network modules. An input module casts the input data to a specific dimension required by the set of network structures. It is a convolutional layer which has an incoming edge from the input data. An output module is a fully connected layer to transform its input to a class output in the one-hot encoding which is a representation of categorical variables as binary vectors. One network structure has exactly one input module and one output module. As for the remaining six modules, their incoming edges and outgoing edges connect to one of the eight network modules.

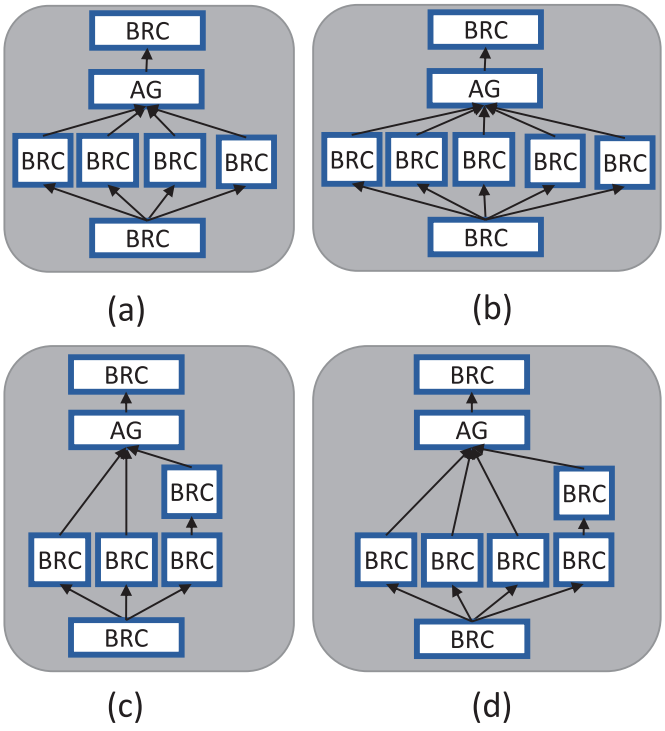

Fig. 1. Local features extraction modules.

A BRC module is composed of a batch normalization layer (Ioffe & Szegedy, 2015), a rectified linear layer (Nair & Hinton, 2010), and a convolutional layer (LeCun et al., 1989), which are connected in sequential order. This composition has been used in CNN (He, Zhang, Ren, & Sun, 2016; Liu et al., 2017b). For the convolution layer, there are four kernels. The size, stride, and padding size of each kernel are 3, 2, and 1 respectively. It is noted that pooling layer is not included in the module because this layer aggregates statistics of features at various locations in the input and downsizes the height and/or width of the feature maps. This layer will constrain the depth of the network. Moreover, modern architectures seldom use pooling layer (Dai, Li, He, & Sun, 2016; He et al., 2016; Yu & Koltun, 2015).

Besides, the total number of incoming edges is not fixed. Therefore, an aggregator module is introduced to combine multiple incoming edges into a single outgoing edge. Since the dimensions of the data from the incoming edges are controlled to be constant, the aggregator module is an addition function that preserves the dimensions in its output.

Lastly, four local features extraction (LFE) modules are introduced. It is composed of several BRC modules and aggregator modules. The local features extraction modules analyze the input and extract the local features in parallel. BRC modules can be connected in parallel to allow the intermediate processing modules to extract multiple features. Refer to Fig. 1a, four BRC modules are connected in parallel to form a parallel part of size four. BRC modules can be connected in series to extract more high-level features. Fig. 1c shows another LFE module, which contains three branches: a parallel part of size two (i.e. the first two branches on the left) and a sequential part of size two (i.e. the rightmost branch). Let *LFE*(*m, n*) denote a LFE module, where *m* and *n* are the size of the parallel part and the size of the sequential part respectively. Hence, Fig. 1a and b can be labeled as LFE(4,0) and LFE(5,0) respectively. LFE(2,2) represents the module construction in Fig. 1c. Lastly, Fig. 1d represents LFE(3,2).

Table 1
Grammar used in ADAG-DNE.

1.1	Start	→ [Input Module →], Topology_1, [→ Output Module]
2.1	Topology	→ [BRC Module →], Topology_2, [→], Topology_3
2.2	Topology	→ LFETopology_1
3.1	LFETopology	→ [LFE(4,0)]
3.2	LFETopology	→ [LFE(5,0)]
3.3	LFETopology	→ [LFE(2,2)]
3.4	LFETopology	→ [LFE(3,2)]

4. Deep neural network topology

In this section, the grammar, the meaning of context, and the translation program are explained.

4.1. Topology grammar

A DNN topology grammar is proposed to represent how DNN modules are assembled. BGBGP searches a set of DNN structures. It generates DNN parse trees from a PCSG which is an extension of a context-free grammar. The grammar in Table 1 restricts the search space for the DNN structures. The 7 rules in the grammar are labeled from 1.1 to 3.4. Terminals are embraced by a pair of square brackets while other are non-terminals. Rule 1.1 defines the starting point and ending point of a network structure, i.e. an input module and an output module. Rule 2.1 says that a network topology *Topology* can be composed of two network topologies. The input to the compositions of network topologies is preprocessed by a BRC module. Rule 2.2 means that a LFE topology (*LFETopology*), which is defined in rules 3.1 to 3.4, is also a network topology. Rule 3.1 specifies a LFE module, where the parameter values of this module are 4 and 0 respectively. Rule 3.2 allows the formation of LFE(5,0). LFE(2,2) can be constructed from rule 3.3. Finally, LFE(3,2) can be constructed from rule 3.4.

The PCSG comprises of a set of rules associated with a Bayesian network for each of them. The Bayesian networks model the probabilistic dependencies among different non-terminals. Because the non-terminals in the grammar represent the topologies, the probabilistic dependencies among different topologies are captured by the Bayesian networks. The details of the derivation can be found in Wong et al. (2014a). In brief, the derivation begins from the rule for *Start* non-terminal on the left-hand side of the arrow. The non-terminals on the right-hand side of the arrow will be derived by one of the rules which have the same non-terminal on the left-hand side of the arrow. The rule is selected following the probability distribution in a Bayesian network. This process repeats until all non-terminals are derived, and a parse tree is formed. Some possible network structures derived from the grammar are shown below:

- Input Module → BRC Module → LFE(4,0) → LFE(5,0) → Output Module
- Input Module → BRC Module → BRC Module → LFE(4,0) → LFE(5,0) → LFE(2,2) → LFE(3,2) → Output Module
- Input Module → LFE(2,2) → Output Module

4.2. Structural context

In Wong et al. (2014a), three context variables are introduced to affect derivation. In the context of this paper, they encode contextual information about the network structures. The first context variable is *Depth*. It is the depth of a node in a parse tree, which is the number of edges from the node to the parse tree's root node. Refer to Fig. 2a, the depth of the shaded node is 1. Fig. 2b shows the corresponding DNN structure represented by the parse tree in Fig. 2a. Depth 1 means the shaded block is in the second layer of

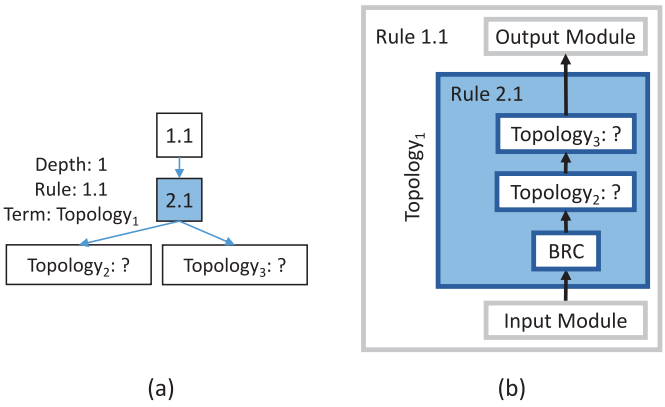

Fig. 2. The meaning of context variables in the parse tree and the corresponding network structure.

boxes in the DNN structure. In general, depth d means the block is in the $(d+1)^{th}$ layer of boxes in the DNN structure.

In addition, *Rule* context variable tells which grammar rule applies leading to the derivation of the current rule. In Fig. 2a, the shaded rectangle representing 2.1 is applied due to rule 1.1 in the grammar, hence the Rule context variable becomes 1.1. Refer to Fig. 2b, this context variable encodes what kind of box (which is formed by rule 1.1) the shaded box is in.

Last but not least, *Term* context variable is about which non-terminal leading to the derivation of the current rule. In the example, rule 2.1 is applied during the derivation of the non-terminal $Topology_1$. This uniquely identifier tells where the rule is applying, or what type of box is using the shaded box in the DNN structure during derivation.

4.3. Translation

After obtaining the parse tree of a DNN structure as described above, it will be converted into Python code. DNN code utilizes the libraries in PyTorch. The code is invoked as a Python module by another main program, which loads the data, trains the network, and evaluates the results. Since the code is provided, this network can be reused in any problem. Unlike Keras (Chollet et al., 2015) adopted in Baldominos et al. (2017), PyTorch (Paszke et al., 2017) improves the flexibility for the researchers to optimize their newly proposed modules in the future.

5. Adaptive grammar-based deep neuroevolution

ADAG-DNE system integrates the components developed in the previous sections into one system. As shown in Fig. 3, it involves five steps.

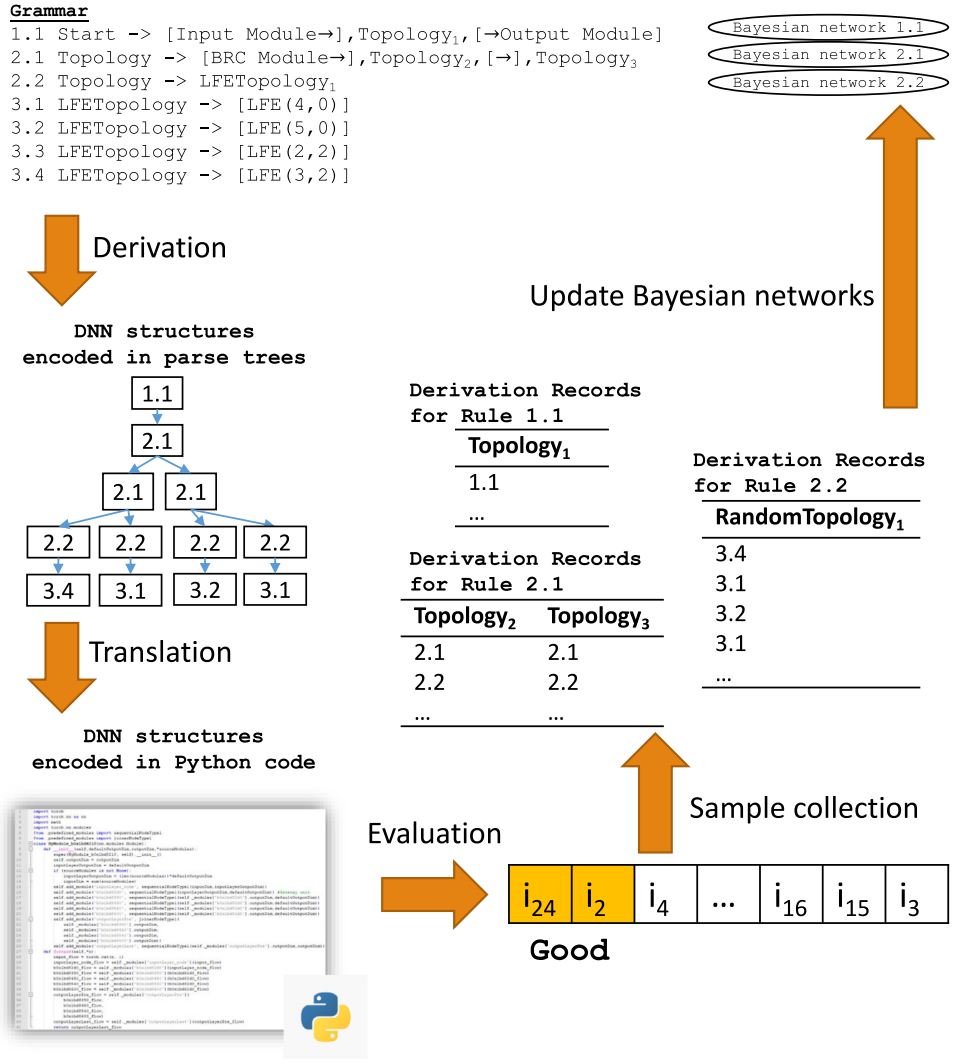

Fig. 3. Adaptive Grammar-based Evolvable (ADAGE) system.

1. Deriving DNN structures encoded in parse trees according to the grammar;
2. Translating the parse trees into Python code;
3. Evaluating the performance of the network using data;
4. Collecting samples from the parse trees of the set of good networks (by their ranking);
5. Updating Bayesian networks in the grammar.

Steps 1–5 are repeated until it reaches the maximum number of generations. The evolutionary system is based on BGBGP system, and the evaluation step relies on PyTorch. Stochastic gradient descent algorithm is adopted to optimize the weights in DNNs.

6. Data set

PhysioNet is a research resource for complex physiological signals (Goldberger et al., 2000). It provides data sets and software for the physiological signal processing. In this paper, a bedside monitor data set collected from four hospitals in the USA and Europe was downloaded. The physiological measures from electrocardiography leads and pulse oximetry were used in the experiment. There are 310 records. Each record lasts 20 s long and contains 5000 features (time points) in total. Our goal is to detect if ventricular tachycardia occurs immediately after 20 s. The records can be categorized into two classes. Patients in ICU who suffered from ventricular tachycardia, which contributes 29% of records. Another class of records were collected from patients in ICU who did not suffer from ventricular tachycardia. Note that the samples were collected from patients in ICU where patients are usually with severe and life-threatening illnesses and injuries. Therefore, the procedure to diagnose ventricular tachycardia can be risky for these patients.

7. Evaluation: comparison with other classifiers

The ADAG-DNE using Depth and Term context variables[1] was adopted using a population of size 30 for 50 generations for 10 runs. The records in the dataset are divided into three parts while maintaining the class distribution to be similar. 30% of records are reserved for testing. Training and validation used 80% and 20% of the remaining records. The training set was used to fit the weights of the network structures. With the validation set, the stopping criteria for the training of a network structure can be adaptable to suppress overfitting. Network structures such that the difference between the training accuracy and the validation accuracy is within 5% were identified. Among them, ten network structures that attained the highest validation accuracy in each run were selected. The selected network structures were run for 10 runs, and the one attaining the highest average validation accuracy was selected as our final model. We now discuss the performance of our final model selected produced from ADAG-DNE. It was compared with the results of neural network methods and non-neural network methods. In our experiment, each method was run for 50 runs. The best model of each method was compared with our evolved network.

[1] The BGBGP system with these context variables attained the lowest amount of fitness evaluations in the royal tree problem when the depth of the parse trees is high as reported in Wong et al. (2014a).

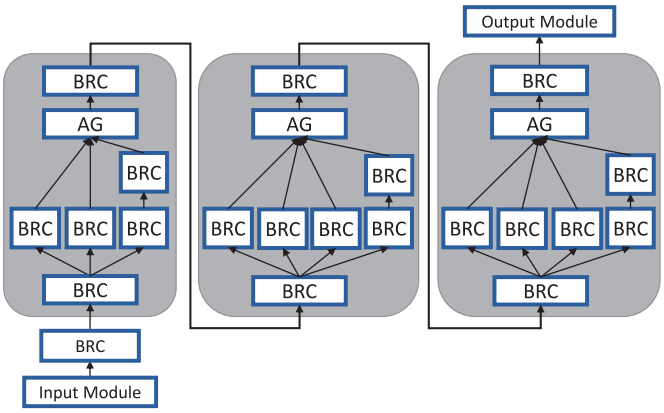

Fig. 4. A network structure discovered by ADAG-DNE system.

The network structure of the best model is shown in Fig. 4. The number of model parameters is around 0.064M using initial learning rate of 0.01 which decays by 10% for every 10 epochs. The network was trained for a maximum of 500 epochs. Six neural networks were chosen to cover a different amount of parameters (from 0.031M to 53M) and network structures. The results are shown in Table 2. ADAG-DNE denotes the best model discovered using our method. AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) and ResNet (He et al., 2016) are state-of-the-art CNNs, which are designed by experts in DNN originally aiming at solving image classification problems. Apart from solving image classification problems, CNN has been applied to signal classification problems as well (OShea, Roy, & Clancy, 2018; Yang, Nguyen, San, Li, & Krishnaswamy, 2015). ResNet introduces a novel architecture with skip connections over some layers by asymptotically approximating the residual functions (He et al., 2016). With this architecture, the training time of ResNet is shorter than that of normal CNN given the same network depth. AlexNet, denoted by AlexNet-A, contains only eight layers: five convolutional layers and three fully connected layers. AlexNet-B differs from AlexNet-A by reducing the number of neurons in the fully connected layers from 4096 to 16, which is the same as that in our evolved network. This dramatically reduces the number of parameters. We can see that ADAG-DNE is better than AlexNet-A and AlexNet-B regarding accuracy but the number of parameters in ADAG-DNE is only 2.2% of that in AlexNet-B. We varied the depth of ResNet such that ResNet10 and ResNet18 have 10 and 18 convolutional layers respectively. The accuracy of ResNet10 is significantly less than ADAG-DNE (i.e. 59% vs 76%). ADAG-DNE is better than ResNet18 regarding accuracy, but the number of parameters used in ADAG-DNE is only 0.05% of that in ResNet18.

Next, two neural networks were created to show that there is no guarantee the performance of the network will increase by merely increasing the number of parameters, since networks with more parameters can represent a more complicated decision boundary. Net-A is made of three fully connected layers in which the number of neurons is 16. There are about 0.016M more parameters in Net-A than ADAG-DNE does, but the performance of Net-A drops by 14%. Net-B connects a convolutional layer to a fully connected layer such that the number of parameters is only half of

Table 2
Results compared with other neural network classifiers.

	ADAG-DNE	AlexNet-A	AlexNet-B	ResNet10	ResNet18	Net-A	Net-B
Parameters	0.064M	24M	2.8M	5.0M	11M	0.080M	0.031M
Accuracy	**76%**	75%	75%	59%	74%	62%	56%

Table 3
Results compared with non-neural network classifiers.

	ADAG-DNE	Logistic Regression	Bayes Network	Naive Bayes	Decision Tree	Rand. Forest	SVM	AdaBoost
Accuracy	**76%**	64%	69%	35%	66%	72%	72%	66%

Table 4
Total amount of DNNs searched.

Max depth	Rand	D	R	T	DR	DT	RT	DRT	Plain
6	11,042	4929	4642	4827	5521	4426	3570	4158	3406
8	11,067	3998	4009	4322	2911	5096	3341	4726	4315

Table 5
Total amount of stable DNNs searched and the retention rate.

Max depth	Rand	D	R	T	DR	DT	RT	DRT	Plain
6	6602	3084	2902	2924	3668	2771	2072	2542	1885
	60%	63%	63%	61%	66%	63%	58%	61%	55%
8	6118	2310	2239	2513	1584	3000	1905	2896	2453
	55%	58%	56%	58%	54%	59%	57%	61%	57%

Table 6
Total amount of stable and predictive DNNs searched and the retention rate.

Max depth	Rand	D	R	T	DR	DT	RT	DRT	Plain
6	4030	1949	1774	1753	2303	1724	1212	1595	1167
	36%	40%	38%	36%	42%	39%	34%	38%	34%
8	3734	1400	1376	1481	984	1894	1170	1690	1455
	34%	35%	34%	34%	34%	37%	35%	36%	34%

that in ADAG-DNE. The performance drops by 20%. This suggests that neuroevolution on the DNN structure is vital to achieve high accuracy using a small number of parameters.

Last but not least, seven well-known non-neural network classifiers, including logistic regression, Bayes network classifier (Heckerman, Geiger, & Chickering, 1995), naive Bayes classifier (John & Langley, 1995), decision tree (Quinlan, 1993), random forest (Breiman, 2001), SVM (Hearst, Dumais, Osuna, Platt, & Scholkopf, 1998), and AdaBoost (Freund, Schapire et al., 1996), are tested. Table 3 shows the results. Only random forest and SVM can achieve over 72% accuracy, whereas ours is 76%. Despite of 4% difference, it is very important in this life-threatening problem. Other classifiers do not perform well. When we compare the results from neural network classifiers and non-neural network classifiers in Tables 2 and 3, ADAG-DNE, AlexNet-A, AlexNet-B, and ResNet18 are better than the non-neural network classifiers by 2–4% regarding accuracy.

8. Evaluation: comparison among different variants

In this section, we analyze the performance of all combinations of context variables in the ADAG-DNE system. Each combination of context variables was executed for 10 runs. The system evolved a population of size 30 for 50 generations for every run. Training accuracy of a network structure is used as its fitness value. The depth-based, rule-based and non-terminal-based context variables are abbreviated by *D, R*, and *T* respectively. For example, the variant *ADAG-DNE/D* means that only depth-based context is used in the Bayesian networks; the variant *ADAG-DNE/DR* means that both depth-based and rule-based context variables are used in the Bayesian networks. If the structural context is not used in the Bayesian networks, we label it as *ADAG-DNE/Plain*. If ADAG-DNE system does not use any Bayesian networks, we label it as *ADAG-DNE/Rand*, i.e. its grammar is not adapted by learning Bayesian networks. This is a GBGP approach for DNE. The other parameters are kept constant in the study unless otherwise specified.

Table 4 shows how many distinct DNNs are searched when the values of maximum depth of parse trees are set to 6 and 8 respectively. Since each configuration executes using a population size of 30 for 50 generations for 10 runs, the maximum number of samples is 15,000 for each configuration. ADAG-DNE/Rand samples 11,042 and 11,067 DNNs at maximum depth 6 and 8 respectively, which are the highest among other settings (Table 4). When using context variables, the system can search less than about 50% of the total DNNs searched by ADAG-DNE/Rand (Table 4).

Observing that many DNNs were generated, but only some of them are good structures. During the evaluation, each DNN was associated with a training accuracy and a validation accuracy. Only if the difference between the training accuracy and the validation accuracy is within 5%, such DNN is regarded as stable. Table 5 shows the total amount and the percentage of the stable DNNs for each configuration. Around 34–45% of DNNs are discarded after the selection because they are not stable.

Apart from the stability, human engineers and domain experts usually want to find a replacement of the existing (generic) classification methods. Random forest classifier and support vector machine are arguably better algorithms (and this is discussed in Section 7). As such, the cut-off validation accuracy is set to 72% using these classifiers as a reference. The following criterion is applied to select a subset from the set of stable networks: the validation accuracy should be at least 72%. The new subset of DNNs is now stable and predictive in Table 6. From the table, ADAG-DNE/Rand discovers 4030 stable and predictive DNNs, which contributes 36% of the whole set, when maximum depth is set to 6. ADAG-DNE/DR discovers slightly more stable and predictive DNNs in term of percentage. This means ADAG-DNE/DR is slightly more efficient. At maximum depth equal to 8, the efficiency of discovering stable and predictive DNNs generated by ADAG-DNE/DR drops by 8%, which is quite high when compared to other configurations.

Table 7
Average accuracies compared among different variants of ADAG-DNE.

Max depth	D	R	T	DR	DT	RT	DRT	Plain
6	73.93%	**74.56%**	74.06%	74.16%	74.19%	73.60%	74.07%	73.86%
8	73.85%	73.92%	73.78%	73.73%	**74.17%**	73.58%	74.04%	74.04%

Table 8
The p-values from Mann–Whitney *U* test on the set of stable and predictive DNNs using the test accuracy.

Max depth	D	R	T	DR	DT	RT	DRT	Plain
6	**4.47** $\times 10^{-2}$	5.84 $\times 10^{-2}$	**3.45** $\times 10^{-3}$	**3.03** $\times 10^{-6}$	**2.30** $\times 10^{-2}$	9.90 $\times 10^{-1}$	1.45 $\times 10^{-1}$	8.10 $\times 10^{-1}$
8	9.69 $\times 10^{-1}$	9.38 $\times 10^{-1}$	9.77 $\times 10^{-1}$	1.00 $\times 10^{0}$	**7.27** $\times 10^{-3}$	9.78 $\times 10^{-1}$	7.68 $\times 10^{-1}$	1.00 $\times 10^{0}$

Since only a threshold is used to define the stable and predictive set, we then compare the classification performance of these networks with that of ADAG-DNE/Rand.

Next, the classification performance of different variants were compared. The best DNN structure in 10 runs for different variants were selected, and then the average test accuracy of these structures from each variant were computed. As shown in Table 7, when the maximum depth is set to 6, ADAG-DNE/R attain the highest accuracy, which is better than that of ADAG-DNE/DR and ADAG-DNE/DT by around 0.04%. When the maximum depth is set to 8, ADAG-DNE/DT attains the highest accuracy (74.17%). Moreover, when maximum depth increases from 6 to 8, the accuracies of some variants decrease. In particular, the accuracy of ADAG-DNE/R decreases by 0.64% while that of ADAG-DNE/DR decreases by 0.43%. ADAG-DNE/DT, ADAG-DNE/RT, and ADAG-DNE/DRT maintain a quite stable performance across different maximum depths.

To quantify the difference, Mann–Whitney *U* test is adopted. It is a nonparametric test comparing two populations that may not be normal distributed, which is observed in our case. Its statistic is the sum of the ranks for observations from one of the samples. Table 8 shows the test on whether using probabilistic dependencies lead to better test accuracy. When the statistics is less than 0.05, the value (in bold) is statistically significant. When the maximum depth is set to 6, ADAG-DNE/D, ADAG-DNE/T, ADAG-DNE/DR, and ADAG-DNE/DT attain values below 0.05. The population of individuals generated by these methods is better than that by ADAG-DNE/Rand. When the maximum depth is increased to 8, only ADAG-DNE/DT can perform statistically better than ADAG-DNE/Rand. The experimental results demonstrate ADAG-DNE/DT helps to evolve stable and predictive DNNs.

Lastly, the evolutionary behaviors of the variants are studied. It is possible to gain insight into why some combination of context variables perform better than the others. Besides, we want to see if it is possible to estimate how many samples of DNN structures are required such that ADAG-DNE/D, ADAG-DNE/T, ADAG-DNE/DR, and ADAG-DNE/DT outperform ADAG-DNE/Rand. To do this, we set an upper limit on how many samples are generated so far in each run. Suppose the upper limit is set to 100, then the *p*-value from Mann–Whitney *U* test statistics of the first 100 DNN structures of each variant are computed. In Fig. 5, the p-value from Mann–Whitney *U* test statistics is plotted against the upper limit.

Concerning the evolutionary behavior when maximum depth is set to 6, the *p*-value tends to decrease as the upper limit increases for all variants, except ADAG-DNE/Plain and ADAG-DNE/RT. If the total number of samples is less than 100, ADAG-DNE/R and ADAG-DNE/DT performs slightly better than ADAG-DNE/Rand. Beyond the upper limit of 100, ADAG-DNE/T and ADAG-DNE/DR are capable of reducing the p-value below 0.05 after the upper limit is set to around 300 respectively. ADAG-DNE/DT and ADAG-DNE/D are less efficient but still reach below 0.05 at around 600 and 900 respectively. At maximum depth 8, only ADAG-DNE/DT is capable of reducing the p-value below 0.05 after collecting 700 samples. It is also observed that there is a small bump before the search can reduce the *p*-value below 0.05. This is the exploration phase in the evolution.

9. Discussion and future work

9.1. Findings and implications

Cardiovascular disease is the leading cause of death among people. Since the problem of detection of abnormal heart rhythm is prevalent, ADAG-DNE provides a viable solution to design suitable CNNs for ECG signal classification. From our results, we found that ADAG-DNE is slightly better than AlexNet-A, AlexNet-B, and ResNet18 regarding accuracy, but the number of parameters used in ADAG-DNE is only 0.05% of that in ResNet18. We also showed that the performance of the learnt model is also better than seven well-known non-neural network classifiers. In terms of clinical application, our proposed solution has the potential to be deployed in ICU to reduce the number of deaths caused by ventricular tachycardia and various heart diseases. This will improve the quality of the health care system in the long run. Since our approach can reduce the size of CNN for physiological signal classification, it is more feasible to deploy our learnt network on a computer chip of health monitoring gadgets than other CNNs tested.

In addition, we have summarized the pros and cons of different classifiers to classify heart rhythms in Table 9. ADAG-DNE offers several distinctive advantages to data scientists and engineers working in the hospitals: 1) Our system does not require much feature engineering and feature selection; 2) Our system does not require hand-crafted CNN structure for the ECG signal classification; 3) The DNN that our system learnt is a CNN. Therefore, the network is invariant to translation; 4) The results of our experiments indicate that the learnt CNN model is simple and effective. The network uses less parameters while achieve a similar or even better performance when compared with other state-of-the-art CNN models.

Theoretically speaking, our results show that there exist some simpler CNN models to attain the same level of classification performance. In other words, there are redundant neurons in some CNN models. Moreover, these simpler and better models can be evolved via evolutionary computation. When the evolved physiological signal classifier is deployed on medical devices, there will be a huge reduction in the computational cost and other operational cost. Furthermore, our novel method using DNE is a generalization of existing CNN methods since our approach does not require hand-crafted CNN structure. We suspect that the hidden

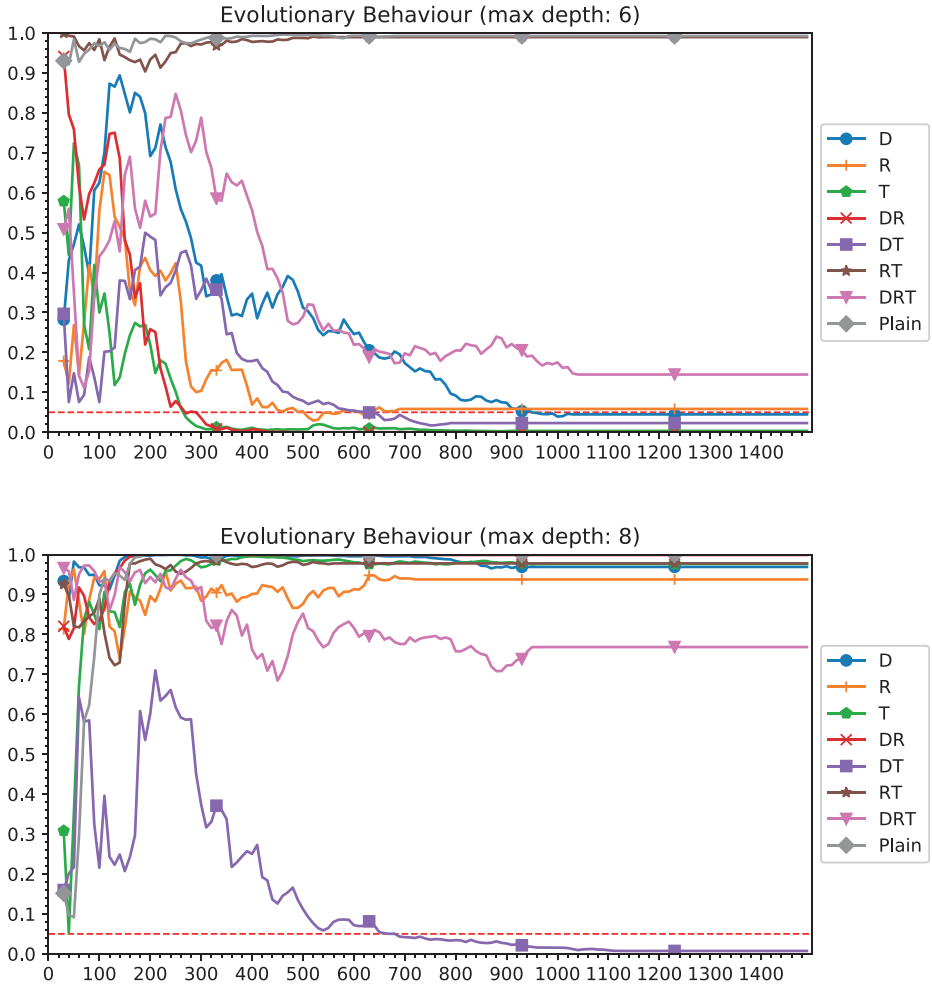

Fig. 5. Evolutionary behavior of our ADAG-DNE system under different configurations. Each graph shows the performance of different configurations of the context variables. The *X*-axis is the upper limit of the number of samples and the *Y*-axis is the *p*-value. Each line represents one configuration of the system.

Table 9
Comparison of selected studies conducted for the detection of abnormal heart rhythm using ECG signals.

Classifier	Computation time	Feature selection	Feature engineering	Handle missing signal features	Design of neural network structure	Model interpretability
Bayesian ANN classifier (Gao, Madden, Chambers, & Lyons, 2005)	Fast	Needed	Hand-crafted	Yes	Manual	Low
Least square SVM (Polat et al., 2008)	Fast	Needed	Via kernel function	No	Not applicable	High
ANN (Lahiri, Kumar, Mishra, Sarkar, & Roy, 2009)	Fast	Needed	Hand-crafted	No	Manual	Low
Pruned and simple KNN (Arif et al., 2012)	Fast	Needed	Hand-crafted	No	Not applicable	High
SVM (Alonso-Atienza et al., 2014)	Fast	Needed	Via kernel function	No	Not applicable	High
Random forest (Masetic & Subasi, 2016)	Fast	Auto-regressive Burg method	Hand-crafted	No	Not applicable	Low
Boosted-CART classifier (Xu et al., 2018)	Fast	Needed	Via adaptive variational mode decomposition	No	Not applicable	High
1-D CNN (Kiranyaz et al., 2016)	Fast with GPGPU	Not required	Learnt automatically	No	Manual	Low
1-D CNN (Zubair et al., 2016)	Fast with GPGPU	Not required	Learnt automatically	No	Manual	Low
1-D CNN (Yıldırım et al., 2018)	Fast with GPGPU	Not required	Learnt automatically	No	Manual	Low
1-D CNN produced by DNE (Our approach)	Fast with GPGPU	Not required	Learnt automatically	No	Automatic	Low

signal patterns, such as magnitude features, of a cardiovascular disease may be captured with a customized multilayer structure of CNN by DNE.

9.2. Limitations and directions

Although the ADAG-DNE method is effective and can reduce the CNN model complexity, CNN in general has some drawbacks. Firstly, training time of CNN can be long, especially when a graphical processing unit is not used in the training process. Secondly, CNN is a black box, meaning that it is difficult to trace a prediction result back to identify which signal features are important. Finally, our current CNN model cannot handle missing feature values due to unrecorded information and noisy measurements.

There are many possible directions for future work. In terms of methodology, our approach can be further improved as the learning rules for training CNN have not be evolved yet. For example, the CNNs generated can be trained using layerwise unsupervised pretraining (Erhan et al., 2010) or curriculum learning (Bengio, Louradour, Collobert, & Weston, 2009). In curriculum learning, a network can be trained in a way such that simpler concepts are learnt first and then higher level concepts are constructed from the simpler ones. Since our ADAG-DNE system can generate various (irregular) forms of CNN structures, we can also study if non-gradient based search techniques, such as CMA-ES (Loshchilov & Hutter, 2016), is a good strategy to learn their parameters.

Next, because many heart diseases share the same risk factors, such as hypertension, diabetes, hyperlipidemia, and obesity (Olafiranye, Jean-Louis, Zizi, Nunes, & Vincent, 2011), ECG signals of different heart diseases may also share similar signal patterns as risk factors. Therefore, another interesting research direction will be to include existing pretrained DNNs (mentioned in Section 2.4) learnt from other relevant datasets in the search space. It is worth investigating whether DNE can intelligently select suitable combinations of networks and construct new structures for the networks. This can utilize existing knowledge and may improve the comprehensibility of the final networks based on the knowledge from other heart diseases. This method may also save time and effort by structurally fine-tuning the DNNs. In the future, our ADAG-DNE method will take advantages of the increasing amount of computational resources to design better networks to solve more significant and challenging problems.

Conflict of interest

No conflict of interest.

Credit authorship contribution statement

Pak-Kan Wong: Conceptualization, Methodology, Software, Data curation, Visualization, Software, Writing - original draft, Writing - review & editing. **Kwong-Sak Leung:** Funding acquisition, Supervision, Writing - review & editing. **Man-Leung Wong:** Conceptualization, Supervision, Project administration, Resources.

Acknowledgments

This research is supported by the Lingnan University Direct Grant DR16A7.

References

Abbass, H. A., Hoai, X., & McKay, R. I. (2002). AntTAG: A new method to compose computer programs using colonies of ants. In *Proceedings of the 2002 congress on evolutionary computation: 2* (pp. 1654–1659). doi:10.1109/CEC.2002.1004490.

Acharya, U. R., Fujita, H., Adam, M., Lih, O. S., Hong, T. J., & Sudarshan, V. K. (2016). Automated characterization of arrhythmias using nonlinear features from tachycardia ECG beats. In *Proceedings of 2016 IEEE international conference on systems, man, and cybernetics* (pp. 533–538). IEEE.

Acharya, U. R., Fujita, H., Oh, S. L., Raghavendra, U., Tan, J. H., Adam, M., et al. (2018). Automated identification of shockable and non-shockable life-threatening ventricular arrhythmias using convolutional neural network. *Future Generation Computer Systems, 79*, 952–959.

Ahmadizar, F., Soltanian, K., AkhlaghianTab, F., & Tsoulos, I. (2015). Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm. *Engineering Applications of Artificial Intelligence, 39*, 1–13.

Alonso-Atienza, F., Morgado, E., Fernandez-Martinez, L., García-Alberola, A., & Rojo-Alvarez, J. L. (2014). Detection of life-threatening arrhythmias using feature selection and support vector machines. *IEEE Transactions on Biomedical Engineering, 61*(3), 832–840.

Amann, A., Tratnig, R., & Unterkofler, K. (2005). Reliability of old and new ventricular fibrillation detection algorithms for automated external defibrillators. *Biomedical Engineering Online, 4*(1), 60.

Arif, M., Malagore, I. A., & Afsar, F. A. (2012). Detection and localization of myocardial infarction using k-nearest neighbor classifier. *Journal of Medical Systems, 36*(1), 279–289.

Assunção, F., Lourenço, N., Machado, P., & Ribeiro, B. (2018). DENSER: Deep evolutionary network structured representation. arXiv:1801.01563.

Baker, T. (2009). Critical care in low-income countries. *Tropical Medicine & International Health, 14*(2), 143–148.

Balasundaram, K., Masse, S., Nair, K., & Umapathy, K. (2013). A classification scheme for ventricular arrhythmias using wavelets analysis. *Medical & Biological Engineering & Computing, 51*(1–2), 153–164.

Baldominos, A., Saez, Y., & Isasi, P. (2017). Evolutionary convolutional neural networks: An application to handwriting recognition. *Neurocomputing*.

Baldominos, A., Saez, Y., & Isasi, P. (2018). Evolutionary design of convolutional neural networks for human activity recognition in sensor-rich environments. *Sensors, 18*(4).

Baumgartner, C. F., Kamnitsas, K., Matthew, J., Smith, S., Kainz, B., & Rueckert, D. (2016). Real-time standard scan plane detection and localisation in fetal ultrasound using fully convolutional neural networks. In *International conference on medical image computing and computer-assisted intervention* (pp. 203–211). Springer.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48). ACM.

Benjamin, E. J., Virani, S. S., Callaway, C. W., Chang, A. R., Cheng, S., Chiuve, S. E., et al. (2018). Heart disease and stroke statistics—2018 update: A report from the American Heart Association. *Circulation*. American Heart Association, Inc.. doi:10.1161/CIR.0000000000000558.

Berbari, E. J., Scherlag, B. J., Hope, R. R., & Lazzara, R. (1978). Recording from the body surface of arrhythmogenic ventricular activity during the st segment. In *American journal of cardiology: 41* (pp. 697–702). Elsevier.

Bradfield, J. S., Boyle, N. G., & Shivkumar, K. (2017). Ventricular arrhythmias. In V. Fuster, R. A. Harrington, J. Narula, & Z. J. Eapen (Eds.), *Hurst's the heart, 14e*. New York, NY: McGraw-Hill Education.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Broomhead, D. S., & Lowe, D. (1988). *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Royal Signals and Radar Establishment.

Chambrin, M.-C. (2001). Alarms in the intensive care unit: How can the number of false alarms be reduced? *Critical Care, 5*(4), 184.

Chambrin, M.-C., Ravaux, P., Calvelo-Aros, D., Jaborska, A., Chopin, C., & Boniface, B. (1999). Multicentric study of monitoring alarms in the adult intensive care unit (ICU): a descriptive analysis. *Intensive Care Medicine, 25*(12), 1360–1366.

Chollet, F., et al. (2015). *Keras*. https://keras.io

Cireşan, D. C., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2013). Mitosis detection in breast cancer histology images with deep neural networks. In *International conference on medical image computing and computer-assisted intervention* (pp. 411–418). Springer.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29* (pp. 379–387). Curran Associates, Inc..

Dasgupta, D., & McGregor, D. R. (1992). Designing application-specific neural networks using the structured genetic algorithm. In *Proceedings of the international workshop on combinations of genetic algorithms and neural networks.* (pp. 87–96). IEEE.

Donchin, Y., & Seagull, F. J. (2002). The hostile environment of the intensive care unit. *Current Opinion in Critical Care, 8*(4), 316–320.

Dzwonczyk, R., Brown, C. G., & Werman, H. A. (1990). The median frequency of the ECG during ventricular fibrillation: Its use in an algorithm for estimating the duration of cardiac arrest. *IEEE Transactions on Biomedical Engineering, 37*(6), 640–646.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*(2), 179–211.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research, 11*(Feb), 625–660.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*(7639), 115.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 13th international conference on machine learning: 96* (pp. 148–156). Bari, Italy.

Gao, D., Madden, M., Chambers, D., & Lyons, G. (2005). Bayesian ANN classifier for ECG arrhythmia diagnostic system: A comparison study. In *Proceedings of 2005 IEEE international joint conference on neural networks, 2005.: 4* (pp. 2383–2388). IEEE.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., et al. (2000). Physiobank, physiotoolkit, and physionet. *Circulation, 101*(23), e215–e220.

Gomez, F., & Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior, 5*(3–4), 317–342.

Gruau, F., & Whitley, D. (1993). Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. *Evolutionary Computation, 1*(3), 213–233.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation, 9*(2), 159–195.

Hasegawa, Y. (2012). Programming with annotated grammar estimation. In S. Ventura (Ed.), *Genetic programming – new approaches and successful applications chapter 3* (pp. 49–74). InTech. doi:10.5772/51662.

Hasegawa, Y., & Iba, H. (2008). A Bayesian network approach to program generation. *IEEE Transactions on Evolutionary Computation, 12*(6), 750–764. doi:10.1109/TEVC.2008.915999.

Hasegawa, Y., & Iba, H. (2009). Latent variable model for estimation of distribution algorithm based on a probabilistic context-free grammar. *IEEE Transactions on Evolutionary Computation, 13*(4), 858–878.

Hauschild, M., & Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation, 1*(3), 111–128.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications, 13*(4), 18–28.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*(3), 197–243.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Igel, C. (2003). Neuroevolution for reinforcement learning using evolution strategies. In *Proceedings of the 2003 congress on evolutionary computation: 4* (pp. 2588–2595). IEEE.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32th international conference on machine learning* (pp. 448–456).

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th conference on uncertainty in artificial intelligence* (pp. 338–345). Morgan Kaufmann Publishers Inc..

Jung, J.-Y., & Reggia, J. A. (2006). Evolutionary design of neural network architectures using a descriptive encoding language. *IEEE Transactions on Evolutionary Computation, 10*(6), 676–688.

Kiranyaz, S., Ince, T., & Gabbouj, M. (2016). Real-time patient-specific ECG classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering, 63*(3), 664–675.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lahiri, T., Kumar, U., Mishra, H., Sarkar, S., & Roy, A. D. (2009). Analysis of ECG signal by chaos principle to help automatic diagnosis of myocardial infarction. *Journal of Scientific & Industrial Research, 68*, 866–870.

Larrañaga, P., & Lozano, J. A. (2001). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Norwell, MA, USA: Kluwer Academic Publishers.

Lawless, S. T. (1994). Crying wolf: False alarms in a pediatric intensive care unit. *Critical Care Medicine, 22*(6), 981–985.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*(4), 541–551.

Liang, J. Z., & Miikkulainen, R. (2015). Evolutionary bilevel optimization for complex control tasks. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation* (pp. 871–878). ACM.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., et al. (2017a). Progressive neural architecture search. arXiv:1712.00559.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017b). A survey of deep neural network architectures and their applications. *Neurocomputing, 234*, 11–26.

Loshchilov, I., & Hutter, F. (2016). Cma-es for hyperparameter optimization of deep neural networks. arXiv:1604.07269.

Martis, R. J., Acharya, U. R., Prasad, H., Chua, C. K., Lim, C. M., & Suri, J. S. (2013). Application of higher order statistics for atrial arrhythmia classification. *Biomedical Signal Processing and Control, 8*(6), 888–900.

Masetic, Z., & Subasi, A. (2016). Congestive heart failure detection using random forest classifier. *Computer Methods and Programs in Biomedicine, 130*, 54–64.

Miller, J. F., & Thomson, P. (2000). Cartesian genetic programming. In *Proceedings of the European conference on genetic programming* (pp. 121–132). Springer.

Millet-Roig, J., Rieta-Ibanez, J., Vilanova, E., Mocholi, A., & Chorro, F. (1999). Time-frequency analysis of a single ECG: to discriminate between ventricular tachycardia and ventricular fibrillation. In *Computers in cardiology 1999. vol. 26 (cat. no. 99ch37004)* (pp. 711–714). IEEE.

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on artificial intelligence: 89* (pp. 762–767). Morgan Kaufmann.

Mühlenbein, H., & Paass, G. (1996). From recombination of genes to the estimation of distributions i. binary parameters. In *Proceedings of the fourth international conference on parallel problem solving from nature* (pp. 178–187). Springer.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning* (pp. 807–814).

Namarvar, H. H., & Shahidi, A. V. (2004). Cardiac arrhythmias predictive detection methods with wavelet-SVD analysis and support vector machines. In *Proceedings of the 26th annual international conference of the IEEE engineering in medicine and biology society: 1* (pp. 365–368). IEEE.

Olafiranye, O., Jean-Louis, G., Zizi, F., Nunes, J., & Vincent, M. (2011). Anxiety and cardiovascular risk: Review of epidemiological and clinical evidence. *Mind & Brain, 2*(1), 32.

O'Neill, M., & Ryan, C. (2003). Grammatical evolution: Evolutionary automatic programming in a arbitrary language. *Genetic Programming, 4*. doi:10.1007/978-1-4615-0447-4.

OShea, T. J., Roy, T., & Clancy, T. C. (2018). Over-the-air deep learning based radio signal classification. *IEEE Journal of Selected Topics in Signal Processing, 12*(1), 168–179.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch.

Pławiak, P. (2018). Novel methodology of cardiac health recognition based on ECG signals and evolutionary-neural system. *Expert Systems with Applications, 92*, 334–349.

Polat, K., Akdemir, B., & Güneş, S. (2008). Computer aided diagnosis of ECG data on the least square support vector machine. *Digital Signal Processing, 18*(1), 25–32.

Quinlan, J. R. (1993). C 4.5: Programs for machine learning. *The Morgan Kaufmann Series in Machine Learning*.

Ratle, A., & Sebag, M. (2001). Avoiding the bloat with stochastic grammar-based genetic programming. In *Selected papers from the 5th European conference on artificial evolution* (pp. 255–266). London, UK: Springer-Verlag.

Rawal, A., & Miikkulainen, R. (2018). From nodes to networks: Evolving recurrent neural networks. arXiv:1803.04439.

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2018). Regularized evolution for image classifier architecture search. arXiv:1802.01548.

Riviello, E. D., Letchford, S., Achieng, L., & Newton, M. W. (2011). Critical care in resource-poor settings: Lessons learned and future directions. *Critical Care Medicine, 39*(4), 860–867.

Salustowicz, R., & Schmidhuber, J. (1997). Probabilistic incremental program evolution. *Evolutionary Computation, 5*(2), 123–141.

Sarfraz, M., Khan, A. A., & Li, F. F. (2014). Using independent component analysis to obtain feature space for reliable ECG arrhythmia classification. In *Proceedings of 2014 IEEE international conference on bioinformatics and biomedicine* (pp. 62–67). IEEE.

Sato, H., Hasegawa, Y., Bollegala, D., & Iba, H. (2012). Probabilistic model building GP with belief propagation. In *Proceedings of the 2012 IEEE congress on evolutionary computation* (pp. 1–8). IEEE.

Shamir, M., Eidelman, L., Floman, Y., Kaplan, L., & Pizov, R. (1999). Pulse oximetry plethysmographic waveform during changes in blood volume. *British Journal of Anaesthesia, 82*(2), 178–181.

Shan, Y., McKay, R. I., Abbass, H. A., & Essam, D. (2003). Program evolution with explicit learning. In *Proceedings of the 2003 congress on evolutionary computation: 3* (pp. 1639–1646). IEEE. doi:10.1109/CEC.2003.1299869.

Soltanian, K., Tab, F. A., Zar, F. A., & Tsoulos, I. (2013). Artificial neural networks generation using grammatical evolution. In *Proceedings of the 21st Iranian conference on electrical engineering* (pp. 1–5). IEEE.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks, 3*(1), 109–118.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines, 8*(2), 131–162.

Suganuma, M., Shirakawa, S., & Nagao, T. (2017). A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the genetic and evolutionary computation conference* (pp. 497–504). ACM.

Tripathy, R. K., Mendez, A. Z., de la O, S., Arrieta Paternina, M. R., Arrieta, J. G., Naik, G. R., et al. (2018). Detection of life threatening ventricular arrhythmia using digital taylor fourier transform. *Frontiers in Physiology, 9*, 722.

Tsoulos, I., Gavrilis, D., & Glavas, E. (2008). Neural network construction and training using grammatical evolution. *Neurocomputing, 72*(1–3), 269–277.

Uther, J., Dennett, C., Duffy, E., Freedman, B., & Tan, A. (1979). Detection of delayed activation potentials of potentials of low amplitude in the vectorcardiogram of patients with recurrent ventricular tachycardia. *Australian and New Zealand journal of medicine*: 9. Adis Press Australasia. 763–763

Whigham, P. A. (1995). Grammatically-based genetic programming. In *Proceedings of the workshop on genetic programming: From theory to real-world applications: 16* (pp. 33–41).

Wong, M. L. (1998). Applying generic genetic programming and cellular encodings to learn fuzzy petri nets. In *Proceedings of the 4th joint conference of information science*.

Wong, M. L., & Leung, K. S. (1995). Applying logic grammars to induce sub-functions in genetic programming. In *Proceedings of the 1995 IEEE conference on evolutionary computation: 2* (pp. 737–740). IEEE. doi:10.1109/ICEC.1995.487477.

Wong, P.-K., Lo, L.-Y., Wong, M.-L., & Leung, K.-S. (2014a). Grammar-based genetic programming with Bayesian network. In *Proceedings of the 2014 IEEE congress on evolutionary computation* (pp. 739–746). IEEE.

Wong, P.-K., Lo, L.-Y., Wong, M.-L., & Leung, K.-S. (2014b). Grammar-based genetic programming with dependence learning and Bayesian network classifier. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation* (pp. 959–966). ACM.

Wong, P.-K., Wong, M.-L., & Leung, K.-S. (2016). Hierarchical knowledge in self-improving grammar-based genetic programming. In *Proceedings of the international conference on parallel problem solving from nature* (pp. 270–280). Springer.

World Health Organization (2003). *Surgical care at the district hospital*. World Health Organization.

Xu, Y., Wang, D., Zhang, W., Ping, P., & Feng, L. (2018). Detection of ventricular tachycardia and fibrillation using adaptive variational mode decomposition and boosted-CART classifier. *Biomedical Signal Processing and Control, 39*, 219–229.

Yang, J., Nguyen, M. N., San, P. P., Li, X. L., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence*.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE, 87*(9), 1423–1447.

Yıldırım, Ö., Pławiak, P., Tan, R.-S., & Acharya, U. R. (2018). Arrhythmia detection using deep convolutional neural network with long duration ECG signals. *Computers in Biology and Medicine, 102*, 411–420.

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *ICLR2016*.

Zubair, M., Kim, J., & Yoon, C. (2016). An automated ECG beat classification system using convolutional neural networks. In *Proceedings of the sixth international conference on it convergence and security* (pp. 1–5). IEEE.