OPTIMIZATION OF RESIDUAL CONVOLUTIONAL NEURAL NETWORK FOR ELECTROCARDIOGRAM CLASSIFICATION

Zeineb Fki
REGIM-Lab.:REsearch Groups in Intelligent Machines
University of Sfax
National Engineering School of Sfax (ENIS)
zeinebfki@ieee.org

Boudour Ammar
REGIM-Lab.:REsearch Groups in Intelligent Machines
University of Sfax
National Engineering School of Sfax (ENIS)
boudour.ammar@ieee.org

Mounir Ben Ayed
REGIM-Lab.:REsearch Groups in Intelligent Machines
University of Sfax
Faculty of Science of Sfax (FSS)
mounir.benayed@ieee.org

December 14, 2021

ABSTRACT

Background and Objective: The interpretation of the electrocardiogram (ECG) gives clinical information and helps in the assessing of the heart function. There are distinct ECG patterns associated with a specific class of arrythmia. The convolutional neural network is actually one of the most applied deep learning algorithms in ECG processing. However, with deep learning models there are many more hyperparameters to tune. Selecting an optimum or best hyperparameter for the convolutional neural network algorithm is challenging. Often, we end up tuning the model manually with different possible range of values until a best fit model is obtained. Automatic hyperparameters tuning using Bayesian optimization (BO) and evolutionary algorithms brings a solution to the harbor manual configuration.

Methods: In this paper, we propose to optimize the Recurrent one Dimensional Convolutional Neural Network model (R-1D-CNN) with two levels. At the first level, a residual convolutional layer and one-dimensional convolutional neural layers are trained to learn patient-specific ECG features over which the multilayer perceptron layers can learn to produce the final class vectors of each input. This level is manual and aims to lower the search space. The second level is automatic and based on proposed algorithm based BO. Results: Our proposed optimized R-1D-CNN architecture is evaluated on two publicly available ECG Datasets. The experimental results display that the proposed algorithm based BO achieves an optimum rate of 99.95%, while the baseline model achieves 99.70% for the MIT-BIH database. Moreover, experiments demonstrate that the proposed architecture fine-tuned with BO achieves a higher accuracy than the other proposed architectures. Conclusions: Our architecture achieves a good result compared to previous works and based on different experiments.

1 Introduction

The ECG is non-invasive electrical recording of the heart. The signal provides a useful information about heart health and can tell more about individuals such as gender, age, biometry and emotion recognition. Researchers have explored this peripheral physiological signal to extract useful markers for future outcome research. Several research works have been achieved in ECG analysis. Challenges have been raised to provide an accurate ECG beats classification. In recent years, Deep Learning (DL) or Deep Neural Network (DNN) is becoming increasingly an important research area. Typically, a DL is a neural network that contains an input layer, successive intermediate layers, namely hidden layers and an output layer. The structure of DL tries to emulate the structure of human brain given a bulky dataset, fast
enough processors and a sophisticated algorithm by constructing layers of artificial neurons that can receive and transmit information. DL offers accurate results with more training data. It is useful for unstructured data. Complex problems can be solved with a greater number of hidden layers. This structure makes possible to continuously adjust and make inferences. It outperforms the traditional machine learning in several applications such as electrocardiogram (ECG) and Electroencephalography (EEG) classification, and more recently industry 4.0 and COVID-19 detection. Recent researches provide many successful algorithms in deep learning. The Convolutional Neural Network (with the acronyms CNN) is actually one of the most applied deep learning algorithms in image recognition including detection of anomalies on ECG. Ebrahimi et al. revealed that the CNN is dominantly found as the appropriate technique for feature extraction, observed in 52% of the studies about explainable deep learning methods for ECG arrhythmia classification. The idea of CNN comes from the biological visual cortex. The cortex consists of small regions that are sensitive to the specific areas of the visual field. Similarly, the CNN builds on small regions inside of an object that perform specific tasks. The algorithm is a hierarchical neural network. It gets the input and processes it through a series of hidden layers. The One-Dimension Convolutional Neural Network (1D-CNN) is a distinguished variant of CNN. It is typically used for time series input with one direction x that represents the time axis. While they have achieved good results in working with a variety of hard problems, the CNNs are usually exposed to overfitting or underfitting problems. Hence, the model fails to predict the output of unseen data or even the output of the training data. In fact, the noise introduced to the input signal slows the learning process. Various types of artifacts could lead to noisy ECG signals such as baseline wander, drift, powerline interference and muscle artifacts. The noisy signals lead to produce high false alarm rates thus the misclassification of ECG beats and misdiagnosis of cardiac arrhythmias. In addition, the ECG signals are non-stationary. Furthermore, the high level of hyperparameters especially in fully connected layers makes the network prone to overfitting. Several previous works have proposed methods to boost classification results based on CNN hyperparameters and regularization. Regularizing the network structure or designing specific training schemes for stable and robust prediction is considered among the hottest topic for efficient and robust pattern recognition in deep learning. A complex model may achieve a high performance on training data since all the inherent relations in seen data are memorized. However, the model is usually unable to perform well for unseen data including validation and test data. In order to solve this issue, different regularization methods were applied in the literature. Xu et al. proposed SparseConnect to alleviate overfitting by sparsifying regularization on dense layers of CNNs. However, they raise, furthermore, the complexity of the model which in turn put the model harder to optimize. In our work, we choose to randomly dropping a few nodes. Unlike conventional methods of tuning based on manual tries to choose the best hyperparameter value, our work proposes to use BO to select an optimum configuration of dropout rate and the number of convolutional layers. In our proposal, two-level process has been established for building a robust Residual 1D-CNN (R-1D-CNN). The level one has the potential of reducing the search space of hyperparameters. The second level allows to test some configurations of the model. The innovative contributions associated with this work can be described as follows:

1. We build a novel R-1D-CNN architecture to detect features of ECG automatically. The proposed architecture presents good performance.
2. To solve the overfitting issue and give robust classification results in real-time through automatic hyper-parameters tuning, we develop an algorithm based on BO.
3. We further explore two datasets for experimental study. Our proposal outperforms another technique of optimization and all the previous works in ECG classification, which displays the performance of our proposed architecture.

The rest of this paper is organized as follows: In Section 2, we outline a short background of the CNN and BO technique. In Section 3, the proposed architecture is detailed. The demonstration and performance of the proposed architecture are indicated in Section 4. At last, Section 5 concludes the paper.

2 Background

2.1 Convolutional Neural Networks

Much of the current research on deep learning has focused on improving and validating existing deep learning algorithms rather developing new algorithms. The CNN is one of the most applied deep learning algorithm. A CNN learns different level of abstraction about an input. CNN performs well in image processing, including image recognition and image classification thanks to its hierarchical layers. The hierarchical property allows to increasingly learning a complex model. For instance, the model learns in the first time the basic elements, then it learns later their parts. Another advantage of CNN is the automatic extraction of feature and it requires minimal pre-processing. The input of CNN is an array of pixels in the format of H x W x D where H= Height, W=Width and D=dimension. The H x W constitute the...
feature map and D is the depth. A grey image of size 32x32 pixels is represented by an array 32X32X1 while an RGB image of the same size is represented by an array of 32X32X3. The structure may include convolutional layers hence its name, pooling layers, Rectified Linear Unit (ReLU) layers and fully connected layers.

- **Convolutional layer**: the main layer of CNN. It consists of a set of filters that exploits the local spatial correlation assuming that near pixels are more correlated than distant pixels. The size of the filter defines the size of each feature map and its depth defines the number of feature maps. All local regions share the same weights called weight sharing. Mathematically saying, a convolution acts as a mixer, mixing two functions to obtain a reduced data space while preserving the information. The model involves training a multilayer architecture without the explicit need of handcrafted input features and is able to extract automatically the features such as edge, blur and sharpen. It helps to remove noise.

- **Pooling layer**: common use is the max-pooling, which implements a sliding window. The max-pooling operation slides over the layer and takes the maximum of each region with a step of stride vertically and horizontally.

- **ReLU**: is a non-linear activation function. It performs a threshold operation. The output takes the same value as the input for the positive values and zero otherwise. The function is used by default for many deep learning algorithms since it performs well and avoids vanishing problem.

- **Fully connected or dense layer**: in a fully connected layer, every neuron is connected to every neuron in the next layer. A model may contain one or more fully connected layer. The dense layer can be the last layer for the classification.

Based on the output, different convolutional dimensions can be used. 1D-CNN is typically used for time series input with one direction x that represents the time axis. Common uses of 1D-CNN are proposed for ECG data classification and anomaly detection [10]. The output shape is one dimension. 2D-CNN performs well for image recognition and classification as the input is an image is of 2 dimensions. The output shape is 2 dimensions. The convolution is calculated based on two directions (x,y). With increasing number of dimensions, CNN 3D applies a three dimensions filter. The filter moves in three directions (x, y, z). The model is helpful in drug discovery [11].

### 2.2 Bayesian Optimization

The effective use of machine learning algorithms is associated with hyperparameters tuning. The hyperparameters adjust the model to a specific database and avoid ongoing training costs. To get up speed on hyperparameters tuning, BO can be used. The technique is based on Bayes' theorem [12] to select the best configuration of hyperparameter values. The Bayes' theorem consists of calculating the conditional probability of an event. Bayes’ theorem uses prior probability distributions to be able to produce posterior probabilities. Prior probability could be the probability of an event before new knowledge is collected. The probability of A conditional on B is defined as [1]

\[
P(A | B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) \cdot P(B | A)}{P(B)}
\]

Where:
- \(P(A)\) = The probability of A occurring
- \(P(B)\) = The probability of B occurring
- \(P(A | B)\) = The probability of A given B
- \(P(B | A)\) = The probability of B given A
- \(P(A \cap B)\) = The probability of both A and B occurring

BO provides a global optimal solution. By limiting the hyperparameters search space on a range of values, the algorithm develops a probabilistic model of the objective function named the surrogate function. Mathematically saying, the algorithm is interested in solving equation 2

\[
x^* = \arg \min_x f(x)
\]

This optimization method takes into account the problem of noise present in the evaluations of equation 3

\[
y = f(x)
\]

Where \(f\) is a black box and expensive to evaluate.

Starting from default parameters e.g. parameter ranges that are used in the literature, the performance evaluation calculated using a numeric score or cost such as the accuracy rate. The aim is selecting a best configuration that
maximize or minimize the cost. The best result achieved by a couple of hyperparameters would be used to construct the tuned model. Hence, the hyperparameters are assigned. For more details about hyperparameter optimization for machine learning models based on BO, please see [13].

### 3 Methods

The proposed classifier is based on CNN algorithm. The first setting of hyperparameters is done manually. The process is iterative to accomplish an acceptable rate of accuracy. We add layers and nodes to the model gradually. The increasing layer number made the manual optimization harder. This configuration is given to the optimization algorithms as the default parameters and runs as the first iteration. By optimizing the neural network loss, the smoothing parameters are optimized to perform the prediction task. A novel R-1D-CNN architecture is presented in our work. The optimization method is described below.

#### 3.1 First Level: Architecture Building

Our proposed CNN at level one was created with 41 plain hidden layers, the first five layers consisting of three convolutional layers, one maxpooling layer and a dropout layer. The block of five layers is repeated seven times with different filter size. The final block of layers consists of three convolutional layers connected to max pooling layer which is followed by another three dense layers. As the network becomes more and more deeper, residual connection [14] is introduced. Hence a new level of depth is appeared. Deep residual learning comes with the benefit of solving the issue involved with vanishing/exploding gradients as well as degradation problem. Residual network achieves this by employing skip connections, or shortcuts to leap across a number of layers. The number of residual blocks in our architecture is fixed to one. The skip connection is located in the position displayed as a residual block in Figure 1. This new architecture allows an efficient training by including skip connections.

#### 3.2 Second Level: Architecture Optimization

To enhance the architecture performance and avoid overfitting issue, we choose to use the Bayesian optimiser. The last is made using Bayesian inference and Gaussian process (GP). This approach is an appropriate algorithm to optimize hyperparameters of classification. By choosing which variables to optimize, and specifying the ranges to search in, the algorithm selects the optimal values. The GP is a well known surrogate model for BO employed for approximating the objective function. It performs well in small dimensional spaces specifically when the number of features meets five features. Table 1 illustrates the selecting hyperparameters, their type and their range. A deep learning model is constructed according to the first level, and the most likely point to be maximized by acquisition function is identified. Some hyperparameters that are very responsive to changes are chosen at first level such as learning rate. BO enables fine-tuning of the model through the regularization of the penalty and determining the optimum number of layers. We choose dropout layer as a technique of regularization.

| Hyperparameter          | Type    | Range   |
|-------------------------|---------|---------|
| Drop Rate               | Real    | 1e-2    |
| Number of Dense Layers  | Integer | 1       |
| Number of Convolutional Layers | Integer | 1       |
| Learning Rate           | Real    | 1e-3    |
| Adam Decay              | Real    | 1e-6    |

Table 1: Hyperparameters end up being run optimization

4 Results

#### 4.1 Setup

We used the python and its data science library to implement our algorithms. We implemented the algorithms using the Keras of TensorFlow library version 2.5 on a Tesla P100 GPU and 25 GB that are provided by Google Colaboratory Notebooks. The training set contains 70% of randomly selected beats and the rest is divided into test and validation set. Each set contains 50% of the remained beats.
4.2 Database

Our proposed model is trained on two publicly available datasets: (1) The MIT-BIH dataset [15] that includes 48 ECG recordings of 30 minutes duration of 47 subjects and 250 sampling rate. Each record is annotated by specialists and can be utilized as ground truth for training, validation and test. The collected data is preprocessed. We build a new dataset that consists of 82813 segments. For beat segmentation, we consider a fixed window multiple of frequency. The raw ECG signal doesn’t require any pre-filtering technique or feature extraction step as used in traditional machine learning algorithms. The database is relatively noise free. Furthermore, the CNN is robust to the noise and features are
automatically extracted during the learning process. (2) The second dataset is 10,000 ECG patients dataset [16]. This dataset consists of 10646 subjects of 10 seconds duration and 500 Hz sampling rate.

4.3 Performance

The performance of our proposed model was evaluated using seven experiments. The first experiment is used to build an architecture to fit MIT-BIH dataset and is achieved 99.70%. The figures 2 and 3 illustrate the accuracy and loss obtained during the training phase. The model runs 100 epochs with early stopped enabled. While the size of the input vector and the number of the hidden layers is large, the model converges in an extremely small time (8 epochs). However, the gap between the validation and the training is significant.

At level two, we introduce the BO. A form of pseudo-code is written to provide precise descriptions of what BO does. The pseudo-code is presented in Algorithm 1. The performance of the algorithm has increased. The BO produced an improvement right after the 13th iteration. The numerical experiments are showing that the resulting accuracy for the optimization with a finite budget outperforms the accuracy of the baseline model.

![Figure 2: Training and Validation Accuracy at level one](image1)

![Figure 3: Training and Validation Loss at level one](image2)

Finally, we build the optimized model for test. The figures 4 and 5 illustrate the accuracy and loss obtained during the training phase. The training accuracy and training loss are respectively close to the validation accuracy and validation loss at second level. The confusion matrix is displayed in Figure 6. Typically, the diagonal elements present the rate of items that are well predicted. Off-diagonal items are mislabelled. The proposed optimized R-1D-CNN properly predicted ECG signals of five distinct classes with a high accuracy of 99.95%. By reviewing the individual performance ratios of the different classes, we notice that the minimum recognition rate belongs to Left bundle branch block class.
Algorithm 1: Algorithm based BO pseudo-code

Begin

Input: Training set \( D = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \} \);
Default hyperparameters set \( H = \{ (h_1, h_2, \ldots, h_d) \} \);
Output: The optimum hyperparameters producing the best classification accuracy

Initialization: \( \text{OptimizationBudget} = 15 \)

\[
\begin{align*}
\text{while } i & \leq \text{OptimizationBudget} \quad // \text{The number of evaluations of fitness function} \\
& \text{do} \\
& \quad \text{gaussianProcessesOptimize}(\text{func} = \text{fitness}, H); \\
& \text{end}
\end{align*}
\]

Function createModel (dropRate, numberDenseLayers, numberConvLayers, learningRate, adamDecay):

\[
\begin{align*}
& \text{model} = \text{newModel}(); \\
& \text{for } i \leftarrow 1 \text{ to numberConvLayers by } 1 \text{ do} \\
& \quad \text{model.add(convLayer)}; \\
& \text{end} \\
& \text{model.add(maxPoolingLayer)}; \\
& \text{model.add(dropOutLayer(dropRate))}; \\
& \text{interimLayerIn} = \text{interimLayerOut} = \text{convLayer}; \\
& \text{for } i \leftarrow 1 \text{ to numberConvLayers by } 1 \text{ do} \\
& \quad \text{interimLayerOut} = \text{convLayer}(\text{interimLayerOut}); \\
& \text{end} \\
& \text{residualLayer} = \text{add}(\text{interimLayerIn}, \text{interimLayerOut}); \\
& \text{model.add(residualLayer)}; \\
& \text{model.add(maxPoolingLayer)}; \\
& \text{model.add(dropOutLayer(dropRate))}; \\
& \text{for } i \leftarrow 1 \text{ to numberConvLayers by } 1 \text{ do} \\
& \quad \text{for } j \leftarrow 1 \text{ to numberConvLayers by } 1 \text{ do} \\
& \quad \quad \text{model.add(convLayer)}; \\
& \quad \text{end} \\
& \quad \text{model.add(maxPoolingLayer)}; \\
& \quad \text{model.add(dropOutLayer(dropRate))}; \\
& \text{end} \\
& \text{for } i \leftarrow 1 \text{ to numberDenseLayers by } 1 \text{ do} \\
& \quad \text{model.add(denseLayer)}; \\
& \text{end} \\
& \text{model.add(denseLayer(5, activation = softmax))}; \\
& \text{opt} = \text{optimizers.Adam(learning_rate = learningRate, decay = adamDecay)}; \\
& \text{model.compile(optimizer = opt, loss = losses.categoricalCrossEntropy, metrics = ['acc'])}; \\
\end{align*}
\]

Function fitness (optimizer = opt, loss = losses.categoricalCrossEntropy, metrics = ['acc']):

\[
\begin{align*}
& \text{model} = \text{createModel}(\text{dropRate, numberDenseLayers, numberConvLayers, learningRate, adamDecay}) ; \\
& \text{result} = \text{model.fit} ; \\
& \text{return} \text{model}
\end{align*}
\]
Table 2: Performance classification using different optimization techniques

| Dataset       | Performance | Without Optimization | With Bayesian | With PSO |
|---------------|-------------|-----------------------|---------------|----------|
| MIT-BIH       | Precision   | 99.7                  | 99.95         | 99.8     |
|               | Recall      | 99.02                 | 99.23         | 99.06    |
|               | F1 score    | 99.35                 | 99.58         | 99.42    |
| 10,000 Patients | Accuracy   | 98.33                 | 99.92         | 99.27    |
|               | Recall      | 98.04                 | 99.22         | 99.15    |
|               | F1 score    | 98.18                 | 99.4          | 99.2     |

with 98%. The highest recognition performance is in the Atrial premature and Normal classes with 100%. The experiments 4-7 aims at testing the model on different databases and using two different optimization techniques. Other metrics are used to evaluate the classification performance. The Recall, the Precision and F1 score can be calculated respectively by Equations (4)–(6)

\[
Sensitivity = Recall = \frac{TP}{TP + FN}
\]

\[
Precision = \frac{TP}{TP + FP}
\]

\[
F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN}
\]

Table 2 display the achieved results. According to the table, the BO outperforms the PSO. The model fit well for the MIT-BIH and 10,000 patients databases.

Figure 4: Training and Validation Accuracy - Optimized model

4.4 Discussion

Both of our architectures at level one and level two present novelties. We demonstrate that the proposed R-1D-CNN architecture, fine tuned by BO method is efficient compared to the state-of-the art architectures. In our experiments, we exploit the R-1D-CNN to classify the ECG signal of two databases. Table 2 displays the classification accuracy of
Figure 5: Training and Validation Loss - Optimized model

Figure 6: Normalized Confusion Matrix for identification of five classes A=Atrial Premature Beat, L=Left bundle branch block beat, N=Normal, R=Right Bundle Branch Block, V=Premature Ventricular Contraction

different architectures. A prior study has shown that ECG signals have successfully learned by AutoEncoder Long Short-Term Memory (AE-LSTM). Other recently developed convolutional architectures have been trained on similar databases. Nevertheless, our proposal achieves the highest accuracy. Compared to the previous works on MIT-BIH database, the CNN architecture provides an automatic extraction of the features, and does not require pre selection feature step. Hence, it is more generic. This attractive feature may facilitate the application of the model in more real-time context, such as the Internet of Things (IoT) data analysis. The results obtained by our architecture are in agreement with those obtained with [22] [21] [23]. Table 4 presents different approaches worked on 1D-CNN and optimized by BO. The table presents mainly the selected hyperparameters, their search space, the obtained optimal values that are used in BO. Our model is able to reach a high classification rate for different databases with a minimum optimization budget and variables. For instance, the drop rate with value 0.010738 and 0.011315 is the optimized value to fit the MIT-BIH database and 10,000 Patients database, respectively.

5 Conclusion

The state-of-arts have produced results in using DL. The last have proven successful in complex problem resolution. Particularly, we explore R-1D-CNN and we propose a novel architecture for ECG analysis. In addition, we develop an algorithm based on BO to give robust classification results in real time through automatic hyperparameters tuning. The experimental results performed on two publicly available ECG Datasets display that the proposed architecture could outperform the state-of-arts architecture. The BO achieves for instance an optimum rate of 99.95%, while the baseline model achieves 99.70% for the MIT-BIH database. Finally, we plan to test the algorithms on other databases, especially
Table 3: Classification accuracy of different architectures

| Reference | Year | Database            | Architecture + Classifier | Accuracy (%) |
|-----------|------|---------------------|---------------------------|--------------|
| [17]      | 2019 | MIT-BIH             | Autoencoder + LSTM        | 99           |
| [18]      | 2020 | MIT-BIH             | Bidirectional LSTM        | 99.64        |
| [19]      | 2021 | MIT-BIH             | GAN+LSTM                  | 99.2         |
| [20]      | 2021 | 10,000 Patients     | CNN                       | 98           |
| The proposed Bayesian-optimized R-1D-CNN | 2021 | MIT-BIH             | R-1D-CNN                  | > 99.9       |
|           |      |                     | and 10,000 Patients       |              |

in dialysis application. Future work will introduce another type of layers and classifiers to control the complexity of the network.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

The research leading to these results has received funding from the Ministry of Higher Education and Scientific Research of Tunisia under the grant agreement number LR11ES48.

References

[1] Jikuo Wang, Xu Qiao, Changchun Liu, Xinpei Wang, YuanYuan Liu, Lianke Yao, and Huan Zhang. Automated ecg classification using a non-local convolutional block attention module. Computer Methods and Programs in Biomedicine, 203, 2021.

[2] Fourati Rahma, Ammar Boudour, Sanchez-Medina Javier, and Alimi Adel M. Unsupervised learning in reservoir computing for eeg-based emotion recognition. IEEE Transactions on Affective Computing, 2020.

[3] Mauricio-Andrés Zamora-Hernandez, John Alejandro Castro-Vargas, Jorge Azorin-Lopez, and Jose Garcia-Rodriguez. Deep learning-based visual control assistant for assembly in industry 4.0. Computers in Industry, 131, 2021.

[4] Mahmood Alzubaidi, Haider Dhia Zubaydi, Ali Abdulqader Bin-Salem, Alaa A Abd-Alrazaq, Arfan Ahmed, and Mowafa Househ. Role of deep learning in early detection of covid-19: Scoping review. Computer Methods and Programs in Biomedicine Update, 1, 2021.

[5] Zahra Ebrahimi, Mohammad Loni, Masoud Daneshtalab, and Arash Gharahbaghi. A review on deep learning methods for eeg arrhythmia classification. Expert Systems with Applications, 7, 2020.

[6] Fan Li, Rui Yan, Reza Mahini, Lai Wei, Zhiqiang Wang, Klaus Mathiak, Rong Liu, and Fengyu Cong. End-to-end sleep staging using convolutional neural network in raw single-channel eeg. Biomedical Signal Processing and Control, 63, 2021.

[7] Dipanjan Moitra and Rakesh Kr. Mandal. Classification of non-small cell lung cancer using one-dimensional convolutional neural network. Expert Systems with Applications, 159, 2020.

[8] Xiao Bai, Xiang Wang, Xianglong Liu, Qiang Liu, Jingkuan Song, Nicu Sebe, and Been Kim. Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments. Pattern Recognition, 120, 2021.

[9] Qi Xu, Ming Zhang, Zonghua Gu, and Gang Pan. Overfitting remedy by sparsifying regularization on fully-connected layers of cnns. Neurocomputing, 328, 2019.

[10] Shumei Chen, Jianbo Yu, and Shijin Wang. One-dimensional convolutional neural network-based active feature extraction for fault detection and diagnosis of industrial processes and its understanding via visualization. ISA Transactions, 2021.
### Table 4: One-dimensional CNN optimized by BO: performance comparison of the proposed method with other methods

| Method | Year | DataBase | Search Space | Range | Optimal Value | n | Epochs | Accuracy (%) | Baseline | Optimum |
|--------|------|----------|--------------|-------|---------------|---|--------|--------------|----------|---------|
| [21]   | 2020 | Landslide data | Number of filters | 4-512 | 117 | 25 | 63 | 80.50 | 83.50 |
|        |      |          | Sequence length | [3, 5, 10, 12] | 10 |     |     |     |     |
|        |      |          | Batch size | 4-128 | 16 |     |     |     |     |
|        |      |          | Activation function | [ReLU, Linear, Tanh, ELU, Sigmoid] | ELU |     |     |     |     |
|        |      |          | Optimization method | [SGD, Adam, Adagrad] | Adagrad |     |     |     |     |
|        |      |          | Neurons in hidden layers | 4-512 | 240 |     |     |     |     |
|        |      |          | Dropout rate | 0-0.8 | 0.66 |     |     |     |     |
|        |      |          | Learning rate | $e^{-5}$ - $e^{-2}$ | $e^{-5}$ |     |     |     |     |
|        |      |          | Batch size | 16-64 | 17 |     |     |     |     |
|        |      |          | Epoch | 20-60 | 54 |     |     |     |     |
|        |      |          | Layer number | 1-5 | 4 |     |     |     |     |
|        |      |          | Conv1 filter number | 6-16 | 10 |     |     |     |     |
|        |      |          | Conv1 filter size | 3-11 | 4 |     |     |     |     |
|        |      |          | Conv2 filter number | 12-32 | 27 |     |     |     |     |
|        |      |          | Conv2 filter size | 3-11 | 11 |     |     |     |     |
|        |      |          | Conv3 filter number | 24-64 | 24 |     |     |     |     |
|        |      |          | Conv3 filter size | 3-11 | 10 |     |     |     |     |
|        |      |          | Conv4 filter number | 48-128 | 66 |     |     |     |     |
|        |      |          | Conv4 filter size | 3-11 | 3 |     |     |     |     |
|        |      |          | Conv5 filter number | 96-256 | - |     |     |     |     |
|        |      |          | Conv5 filter size | 3-11 | - |     |     |     |     |
|        |      |          | Number of dense layers | 1-6 | 1/6 | 15 | 8 | 99.70 | >99.9 |
|        |      |          | Number of convolutional layers | 1-6 | 3/2 |     |     |     |     |
|        |      |          | Learning rate | $e^{-5}$ - $e^{-4}$ | $0.0000006$ |     |     |     |     |
|        |      |          | Adam decay | $e^{-5}$ | $0.011315$ | 0.010738 | 0.011315 |

[11] Mohammad Ali Rezaei, Yanjun Li, Dapeng Wu, Xiaolin Li, and Chenglong Li. Deep learning in drug design: Protein-ligand binding affinity prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[12] Daniel Berrar. Bayes’ theorem and naive bayes classifier. In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*. 2019.

[13] A. Helen Victoria and G. Maragatham. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 3, 2020.
[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[15] G.B. Moody and R.G. Mark. The impact of the mit-bih arrhythmia database. IEEE Engineering in Medicine and Biology Magazine, 20, 2001.

[16] Zheng Jianwei, Zhang Jianming, Danioko Sidy, Yao Hai, Guo Hangyuan, and Rakovski Cyril. A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients. Scientific Data, 7, 2020.

[17] Ozal Yildirim, Ulas Baran Baloglu, Ru-San Tan, Edward J. Ciaccio, and U. Rajendra Acharya. A new approach for arrhythmia classification using deep coded features and lstm networks. Computer Methods and Programs in Biomedicine, 176, 2019.

[18] Siti Nurmaini, Alexander Edo Tondas, Annisa Darmawahyuni, Muhammad Naufal Rachmatullah, Jannes Effendi, Firdaus Firdaus, and Bambang Tutuko. Electrocardiogram signal classification for automated delineation using bidirectional long short-term memory. Informatics in Medicine Unlocked, 22, 2021.

[19] Maher Ibrahim Sameen, Biswajeet Pradhan, and Saro Lee. Application of convolutional neural networks featuring bayesian optimization for landslide susceptibility assessment. Catena, 186, 2020.

[20] Fatma Murat, Ozal Yildirim, Muhammed Talo, Yakup Demir, Ru-San Tan, Edward J. Ciaccio, and U. Rajendra Acharya. Exploring deep features and ecg attributes to detect cardiac rhythm classes. Knowledge-Based Systems, 232, 2021.

[21] Dachuan Shi, Yunguang Ye, Marco Gillwald, and Markus Hecht. Designing a lightweight 1d convolutional neural network with bayesian optimization for wheel flat detection using carbody accelerations. International Journal of Rail Transportation, 2020.

[22] Mohammed Gamal Ragab, Said Jadid Abdulkadir, Norshakirah Aziz, Hitham Alhussian, Abubakar Bala, and Alawi Alqushaibi. An ensemble one dimensional convolutional neural network with bayesian optimization for environmental sound classification. Applied Sciences, 11, 2021.