Evolutionary Intelligence
https://doi.org/10.1007/s12065-022-00758-3

RESEARCH PAPER

Diverse activation functions based-hybrid RBF-ELM neural network for medical classification

Roguia Siouda[1] · **Mohamed Nemissi**[1] · **Hamid Seridi**[1]

Received: 21 September 2021 / Revised: 30 May 2022 / Accepted: 4 July 2022

Abstract
The current abundance of the collected biomedical data provides an important tool for the development of medical data classification systems. However, processing big data requires powerful algorithms. In this perspective, we propose a hybrid classifier that combines radial basis function (RBF) and extreme learning machine (ELM) neural networks. This combination is motivated by the high performances and the complementary of these two types of neural networks. The basic idea relies on complementing a compact RBF network by an ELM network that contains a diversity of hidden neurons. The optimization of the number, forms, and types of the ELM hidden neurons is performed using a genetic algorithm (GA). The objectives of the proposed classifier can be summarized as follows. First, it benefits from the complementary properties of RBF and ELM, like local response of RBFs and global response of ELM. Second, it makes use of the advantages of ELM, like fast training and the possibility of using a variety of activation functions. Third, it alleviates the ill conditioning problem of ELM by joining the systematic initialization of RBF to the random initialization of ELM. Fourth, the optimization process, performed using GA, is simplified because it concerns only the added neurons, which their role is complementing the RBF network. To assess the performance of the proposed classifier, we carry out tests on six medical datasets from the UCI machine learning repository and compare the obtained results with those of other state-of-the-art works. The obtained average performance measurement, i.e., accuracy, sensitivity, and specificity for Wisconsin breast cancer are 97.38%, 98.38%, 96.85%, for Pima Indians diabetes are 77.61%, 57.35%, 88.22%, for heart Statlog are 83.71%, 77.92%, 88.34%, for hepatitis are 87.10%, 95.89%, 40.10%, for Parkinson are 92.62%, 96.50%, 80.76%, and for liver-disorders are 72.48%, 82.68%, 58.39% respectively

Keywords RBF neural network · Extreme learning machine · Genetic algorithm · K-means · Classification · Medical diagnosis

1 Introduction

In the past few years, medical informatics research and production have been growing in an increasing rate. With the modern technological progressions, the collected biomedical data has also been growing in an increasing rate. They include a large variety of types, like blood tests, imaging data, biomedical signals, patients' records ... etc. On one hand, these big data provide useful information to design more powerful analyzing and aid-decision systems. On the other hand, handling such amount of data generally presents enormous challenges as regards to their size, complexity, multidimensionality, heterogeneity and incompleteness. Consequently, this requires the development of advanced pattern recognition methods that can face these challenges. Neural networks have gained popularity as efficient computational approaches in this field because of their learning and approximation abilities. In this work, we consider two important types of neural networks: RBF and ELM models, which have been successfully applied to many medical systems.

Radial basis function neural networks (RBFNNs) were introduced in 1988 by by Broomhead and Lowe [1]. They are motivated by the local reaction observed in some biologic neurons, so they include special hidden neurons with local

✉ Roguia Siouda
siouda.roguia@univ-guelma.dz

Mohamed Nemissi
nemissi.mohamed@univ-guelma.dz

Hamid Seridi
seridi.hamid@univ-guelma.dz

[1] LabSTIC Laboratory, University of 8 Mai 1945, PB 401, Guelma, Algeria

Published online: 20 July 2022

response (RBFs). The implementation of RBFNN consists in designing three important components: (i) the structure, i.e. the number of RBFs; (ii) the parameters of RBFs, i.e. their centers and widths; and (iii) the output weights. Choosing the accurate structure of the network is generally the main difficulty facing the developers of neural networks. This problem is more important in the case of RBFNN where the hidden neurons must provide good coverage of the input space. To deal with this issue, two main strategies have been adopted in the literature. In the first strategy, the structure is determined in an independent phase, and then the remaining parameters are accordingly adjusted. In the second strategy, the structure of the network and the other parameters are all adjusted during the training process. The recent works, whatever based on the first or second strategy, are largely based on bioinspired metaheuristic algorithms. For example, in [2], the authors proposed a hybrid RBFNN based on Fuzzy C-Means (FCM) and polynomial neural networks in order to enhance the discriminant capabilities. In this system, genetic algorithm (GA) is used to optimize the main parameters of the model, such as the fuzzification coefficient and the number of input polynomial fuzzy neurons. In [3], the authors proposed to design RBFNN using a cooperative particle swarm optimization (CPSO). It consists of two distinct swarms. The first swarm calculates the network structure and the centers using a non-symmetric fuzzy means algorithm, while the second calculates the widths. In this system, the two swarms cooperate by exchanging information. In [4], the authors proposed an RBFNN based on bee-inspired algorithm (cOptBees). In this system, a bee-inspired clustering algorithm is used to determine the number and the centers of RBFs. Next, the widths are determined using the distribution of the clustered data and the output weights are calculated using pseudo-inverse method. In [5], the authors proposed an RBFNN based on K-means and BAT-inspired optimization algorithm. In this system, the K-means is used with cluster validity index to define the centers of the RBFs and the BAT algorithm is used to define the output weights. In [6], the authors proposed an eigenvector-based clustering method to calculate the RBF centers. This method is based on calculating the principal components of the data matrix instead of the iterative calculation process of K-means clustering. Subsequently, the output weights can be calculated via either pseudo-inverse solution or the gradient descent algorithm. In [7], the authors proposed to use the Biogeography-based optimization (BBO) algorithm for training an RBFNN. In this system, the number of RBFs is fixed at the beginning and the remaining parameters, i.e. centers, widths and output weights are optimized simultaneously. In [8], the authors aimed at designing a robust RBFNN based on self-organizing map (SOM) networks. In this model, the SOM-based clustering algorithm determines the centers of RBFs while P-nearest neighbors heuristic method determines their widths. In order to enhance the robustness of the proposed system, the authors proposed adding independent random Gaussian noise to the training samples. In [9], the authors proposed to design an RBFNN classifier in three stages. In this model, K-means clustering method first defines the centers of RBFs, then PSO optimizes the widths of RBFs and, finally, the back-propagation algorithm is used for fine-tuning all parameters, i.e. centers, widths and output weights. In [10], the authors proposed a deep RBF neural system for medical classification based on the deep autoencoder. In this system, the autoencoder is used to decrease the number of features of the presented samples. The obtained new features are then presented to the RBF neural network.

On the other hand, an interesting learning algorithm for single hidden feed-forward neural networks (SHFFNNs) called extreme learning machine (ELM) has been recently emerged [11]. It is based on random initialization of the input weights and biases and analytic calculation of the output weights. Compared to the traditional gradient descent algorithms, ELM is much faster and has better generalization ability. Furthermore, it can be performed with less human intervention, like setting the stopping criteria, learning rate and number of epochs. However, ELM suffers from two main drawbacks: it is sensitive to the initial input weights and it requires a large number of hidden neurons. A variety of methods have been proposed to tackle these problems. Among these methods, metaheuristic-based approaches have had great interest because of their efficiency in solving complex problems. These approaches include PSO [12, 13], GA [14, 15], cuckoo search [16] and deferential evolution [17].

RBF and ELM neural networks have been combined in order to introduce more powerful models. The combination of RBF and ELM has been performed according to two main approaches. The first approach was introduced in [18]. The resultant model is a single layer network with hidden RBF neurons. In this model, the centers and width of RBFs are randomly initialized and the output weights are analytically calculated using pseudo inverse method. Several methods have been used to improve this model, for example, micro-genetic algorithm [19]; affinity propagation clustering and Laplacian ELM [20]; simulated annealing algorithm [21]. The second approach assembles RBF and ELM in a cascade. The resultant model has two hidden layers. In this model, the outputs of the RBF constitute the input of the ELM [22, 23].

Considering the above discussion, we can note that the developers of RBF, ELM and even RBF-ELM are mostly facing the same problem: defining the proper structure. Metaheuristic-based approaches have been successfully used to optimize the structure of these two types of neural networks. Nevertheless, optimizing the structure and the parameters at the same time is very hard and time-consuming. This may result in using small populations and consequently can lead to lower diversity and poor performance. In this

work, we deal with this problem in a completely different way. We propose a hybrid model that consists of an RBFNN with minimal structure complemented by an ELMNN with a diversity of activation functions. The number, types and forms of the added neurons are optimized using GA. Therefore, we combine the advantages and the complementary of RBFNNs and ELMNNs to overcome their problems, namely defining the appropriate structure. Furthermore, the optimization process in the proposed classifier is considerably simplified because it concerns only the added neurons, which their role is just to complement the RBFNN.

The rest of this paper is organized as follows. Section 2 recalls some background concerning the RBFNNs and ELM. Section 3 describes the principles of the proposed model and discusses its implementation. Section 4 presents and analyses the experimental results conducted on six biomedical datasets. Finally, Sect. 5 concludes this paper.

2 Theoretical background

This section introduces some theoretical principles of the RBF and ELM neural networks, which are used in this work.

2.1 Radial Basis Function Neuronal Networks (RBFNNs)

An RBF neural network is a feed-forward network with a single hidden layer [24]. In these networks, the activation functions used in the hidden neurons are radial basis functions (RBFs) rather than sigmoid functions. This kind of activation function is inspired by the local response observed in some biologic neurons. Therefore, the RBFs have local response and their values depend only on the distance from their centers. Figure 1 illustrates an example of RBFNNs. In this example, the network has N input neurons, M hidden neurons and J output neurons.

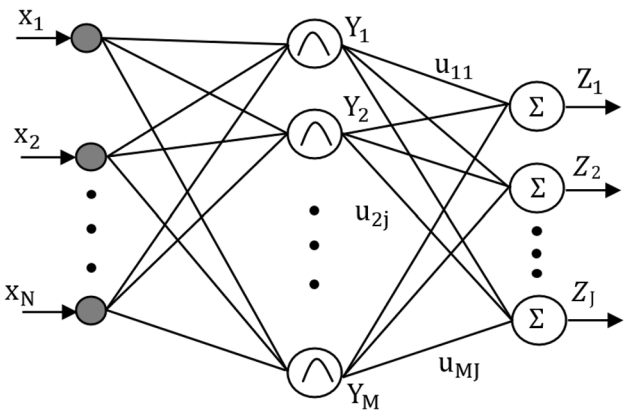


Fig. 1 An example of RBF neural network with N–M–J architecture

In the classification problems, the number of input neurons is equal to the input dimension and the number of output neurons is equal to the number of classes.

In this work, we use the Gaussian functions, which are the most commonly used. The outputs of the hidden layer are then given by:

$$Y_m(X) = \exp\left(-\frac{\| X - V_m \|^2}{2\sigma_m^2}\right) \tag{1}$$

where $X = (x_1, x_2 \ldots x_N)$ is the input vector, $V_m = (v_{1m}, v_{2m} \ldots v_{Nm})$ and σ_m are, respectively, the center vector and width corresponding to the m^{th} hidden neuron.

The network outputs are a linear weighted combination of the hidden output. The j^{th} output Z_j is given by:

$$Z_j = \sum_{m=1}^{M} U_{mj}.Y_m(X) \tag{2}$$

where M is the number of hidden neurons, Y_m is the output of the m^{th} hidden neuron and U_{mj} is the weight connecting the m^{th} hidden neuron and the j^{th} output.

2.2 Extreme learning machine (ELM) algorithm

Extreme learning machine is a learning algorithm for single hidden layer feed-forward neural networks [11]. In this algorithm, the input weights and biases are randomly initialized while the output weights are analytically determined in one-step stage using pseudo-inverse solution. This makes this algorithm extremely fast compared to the traditional gradient descent algorithms.

Figure 2 illustrates the structure of an ELM based-neural network.

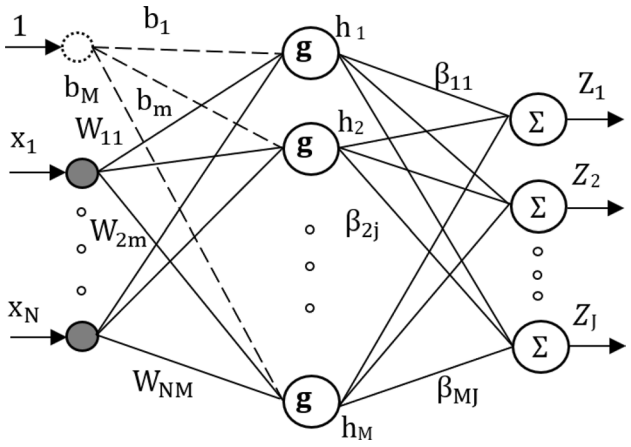


Fig. 2 An example of ELM based-neural network with N inputs, M hidden neurons and J output neurons

For Q distinct samples $\{(x_q, t_q), q = 1 : Q\}$ where $x_q = [x_{q1}, x_{q2}, \ldots, x_{qN}]^T \in \mathbb{R}^N$ and $t_q = [t_{q1}, t_{q2}, \ldots, t_{qJ}]^T \in \mathbb{R}^J$.

An ELMNN with M hidden neurons and activation function $g(x)$ is represented by [11] :

$$\sum_{m=1}^{M} \beta_m . G_m(x_q) = \sum_{m=1}^{M} \beta_m . G(w_m . x_q + b_m) = t_q \tag{3}$$

where $w_m = [w_{m1}, w_{m2}, \ldots, w_{mN}]^T$ is the input weight connecting the input layer to the m^{th} hidden neuron and b_m is the bias of the m^{th} hidden neuron.

The output equation can be written as:

$$H\beta = T \tag{4}$$

where

$$H = \begin{bmatrix} g(w_1 . x_1 + b_1) & \cdots & g(w_M . x_1 + b_M) \\ \vdots & \cdots & \vdots \\ g(w_1 . x_Q + b_1) & \cdots & g(w_M . x_Q + b_M) \end{bmatrix}_{Q \times M},$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \vdots \\ \beta_M^T \end{bmatrix}_{M \times J} \quad and \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ \vdots \\ t_Q^T \end{bmatrix}_{Q \times J} \tag{5}$$

Here, $\beta_m = [\beta_{m1}, \beta_{m2}, \ldots, \beta_{mJ}]^T$ is the optimal weight connecting the m^{th} hidden neuron to the output layer. These output weights can be analytically determined by the minimum norm of the least square solution:

$$\beta = H^{\ddagger} T \tag{6}$$

where $H^{\ddagger}$ is the Moore–Penrose generalized inverse of the matrix H.

To enhance the generalization performances of ELM and its robustness with respect to outliers, a regularized version was introduced [25]. The output weight β is then estimated by the following equation:

$$\beta = H^T \left(\frac{I}{C} + HH^T \right)^{-1} T \tag{7}$$

where C is the regularization factor ($C > 0$) and I is the unit matrix.

3 The proposed classifier

In this work, we introduce a hybrid neural classifier that combines RBF and ELM neural networks. In the following subsections, we describe in detail the main idea and motivations behind the proposed system and its architecture and training process.

3.1 Main idea and motivations

The key idea of the proposed classifier relies on making use of the complementary properties of RBFNN and ELMNN in order to provide better performances with an accurate structure. The proposed model is based on an RBFNN with minimal architecture complemented by an ELMNN with a diversity of activation functions. The number of the added neurons and the types and forms of their activation functions are optimized using GA. We opted to use GA as they can easily model a hierarchical structure and combine binary and real encoding [26, 27]. The binary encoding represents the model structure while the real encoding represents the model parameters. In our model, the binary encoding is used to define the number of added neurons and the integer and real encoding are used to define the types and forms of the added neurons.

The motivations behind the proposed approach are as follows:

First, the hidden neurons in RBFNNs are based on local responses while in ELMNNs they are based on global responses. Therefore, by their combination, we can have better decisions boundaries with a smaller number of hidden neurons.

Second, the initial weights in ELMNNs are randomly initialized which may lead to ill-conditioned problem, especially in the case of a large number of hidden neurons [28, 29]. Contrary, RBFNN can be initialized using prior knowledge by applying clustering methods on the training data. Therefore, by combining both types, we can prevent ill-conditioning problems. Furthermore, this combination reduces the dependency of ELMNN to the random initial weights.

Third, it was reported in several works [30–32] that a network with different types of activation functions has better generalization capacities. Even for deep neural networks, several approaches have been proposed for replacing the common ReLU activation functions [33, 34]. It was also reported that an ELMNN with large variety of activation functions can distinguish between disconnected regions of any form [35, 36]. Therefore, we propose adding neurons with different types of activation functions and using GA to optimize their types and forms. Furthermore, the GA is used to define the optimum number of the added neurons. This way, we aim at overcoming the problem of defining the proper structure, which constitutes a considerable problem for both types of neural networks.

Fourth, evolving neural networks using meta-heuristic approaches is generally complex and time-consuming especially in the case of optimizing both the structure and the parameters. In this work, we propose to optimize the additional neurons only. Since the role of these neurons is

just to complement the RBFs, the optimizing process can be effectively alleviated.

Fifth, in the proposed model, the outputs of the RBFs and those of the added neurons are concatenated and considered as the output of the hidden layer of an ordinary ELM. The output weights are then calculated in one-step phase using the pseudo-inverse method. In general, updating the output weights is simpler than updating the hidden and inputs ones [37]. Therefore, we make use of the fast training of ELM to simplify the calculation of the GA cost function.

3.2 Architecture

The proposed model is a single hidden layer feed-forward neural network that consists of two parts. The architecture of the proposed classifier is depicted in Fig. 3.

The first part, RBFNN, consists of the RBFs and their corresponding input weights. The role of these hidden neurons is to measure the distance between the presented sample and their centers. The weights connecting the inputs to these neurons are equal to '1'.

The second part, ELMNN, consists of the added neurons and their corresponding input weights. In this part, the weights are randomly initialized and the hidden neurons have general activation functions.

In this work, we use four types of the most commonly used activation functions, given as follows [35, 36]:

1. Sigmoid:

$$Hs(x) = \frac{1}{1 + \exp\left(-(a.x + b)\right)} \tag{8}$$

2. Gaussian:

$$Hg(x) = \exp\left(-b \parallel x - a \parallel^2\right) \tag{9}$$

3. Hard-limit:

$$Hl(x) = \begin{cases} 1, & if(a.x - b) \geq 0 \\ 0, & otherwise \end{cases} \tag{10}$$

4. Multi-quadratic:

$$Hq(x) = \left(\parallel x - a \parallel^2 + b^2\right)^{\frac{1}{2}} \tag{11}$$

where x is the input features, a is the slope coefficient and b is the bias coefficient.

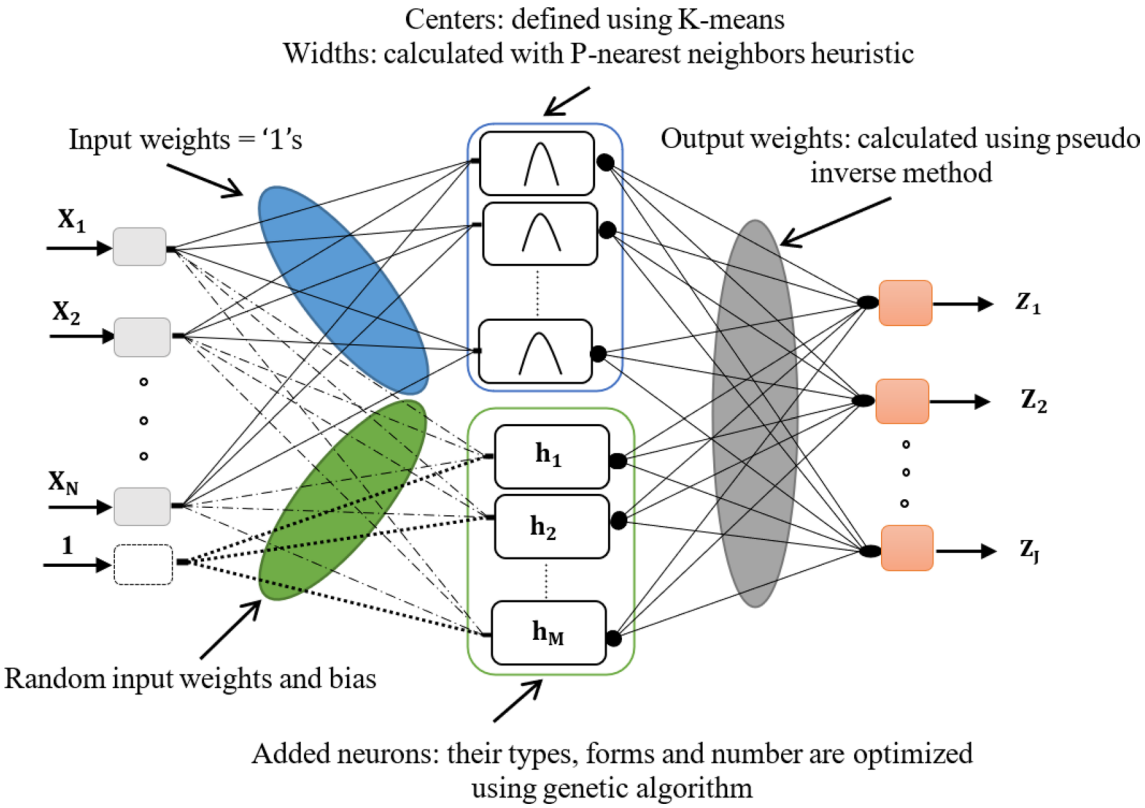


Fig. 3 The structure of the proposed neural classifier RBF-ELM-GA. It has one hidden layer that contains two set of neurons: initial RBFs and added neurons. The latter have different types and forms optimized using GA

In order to obtain more variety in the added neurons, we propose using different forms of the above functions. More specifically, we use different values of the parameters a and b in each activation function. The outputs of the RBFs and the added hidden neurons are concatenated to form a single vector. This vector is considered as the output of the hidden layer in a conventional ELMNN.

3.3 Training process

The development of the proposed classifier is performed in two stages: defining RBFs and adding neurons Fig. 4.

3.3.1 Defining RBFs

In this stage, K-means clustering method is first used to define the centers of RBFs. In general, the clustering methods are unsupervised algorithms applied to unlabeled samples. In this work, we apply the clustering to each class separately in order to have more accurate RBFs. Next, the P-nearest neighbors' method [38] is used to calculate the widths of RBFs. The width of the m^{th} hidden neuron is given by:

$$\sigma_m = \frac{1}{2}\left(\sum_{n=1}^{P} \| V_n - V_m \|^2\right)^{\frac{1}{2}} \tag{12}$$

where $V_n (n = 1 \ldots P)$ are the P-nearest neighbours' of the centre V_m.

3.3.2 Optimizing the added neurons using GA

In the second stage, the number, the types and the forms of the added neurons are optimized using GA. This process is given as follows:

3.3.2.1 Coding The chromosome coding is the most important step in GA algorithm as it defines how to represent the solutions in a set of genes. In this work, each chromosome consists of two parts: control and parameters genes Fig. 5.

In the first part, each bit corresponds to one of the hidden neurons. This part is then formed by a sequence of bits where the values '1' indicate that the corresponding hidden neurons are activated and those corresponding to '0' are inactivated. The sum of the values '1' represents the number of the added neurons.

The second part of chromosome represents the parameter genes. In this part, each hidden neuron is represented by three

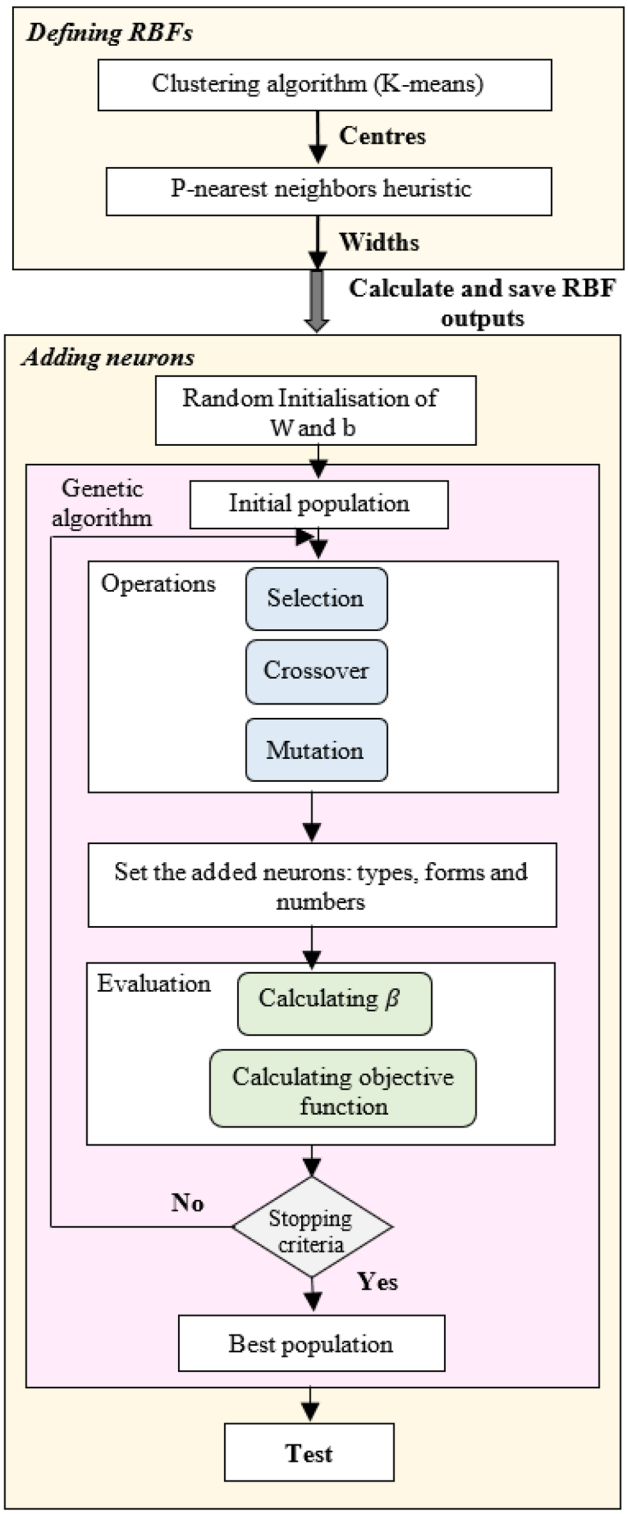


Fig. 4 The general scheme of the proposed classifier: it is composed of two main stages: defining RBFs and adding neurons using GA

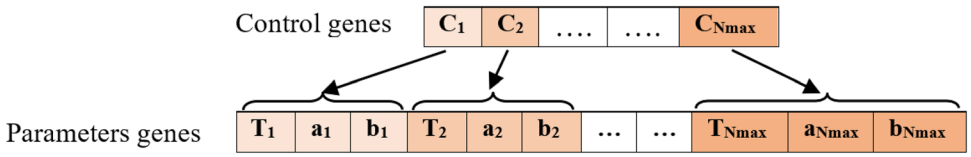


Fig. 5 General representation of chromosome coding

bits. The first bit is an integer with values $\{2, 3, 4, and 5\}$ that defines the type of the activation function. The two other bits are real numbers that represent the parameters of the activation function. The proposed GA includes three types: binary, integer and real values.

Figure 5 illustrates the general representation of chromosome coding. T_i represents the type of the activation function (Sigmoid, Gaussian, Hard-limit and Multi-quadratic) corresponding to the i^{th} hidden neuron. a_i and b_i are the first and second parameters of the activation function corresponding to the i^{th} hidden neuron.

Figure 6 gives an example of chromosome coding. In this example, the maximum number of added neurons is four. Among them, only the second and the fourth are activated. The types of the four neurons are respectively: Sigmoid, Hard-limit, Gaussian and Multi-quadratic.

3.3.2.2 Fitness function The role of GA in this work is to define a set of neurons with different types and forms providing, not only, a minimum Root Mean Square Error, but also with the minimum number. Therefore, we use a multi-objective fitness function that considers both specifications. It is given by:

$$Fitnessfunction = RMSE + (C * ComNetwork) \tag{13}$$

where C is a constant that balances the effect of RMSE and ComNetwork.

RMSE is the mean of the squared differences between the actual and the predicted values for all samples. It is given by:

$$RMSE = \sqrt{\frac{1}{Q}\sum_{q=1}^{Q}\left(Z_q - T_q\right)^2} \tag{14}$$

where Q is the number of the training samples; Z_q and T_q are the estimated and actual classes corresponding to the q^{th} sample.

ComNetwork represents the complexity of the network. It is given by the proportion of the number of hidden neurons, M , to the maximum number of neurons, Nmax.

$$ComNetwork = \frac{M}{Nmax} \tag{15}$$

3.3.2.3 Genetic operators The genetic operators play a crucial role in evolving the populations and, consequently, in the overall performance. There are three basic operators: selection, crossover and mutation. In this work, we use the most commonly used operators.

- □ For the selection, we use the roulette wheel method.
- □ For the crossover, we use the double-point method as our chromosomes include two parts (control and parameters). Therefore, two locations are randomly selected: The first one is applied to the control genes and the second to the parameter genes. This way, the new individuals can inherit both information from the parents.
- □ For the mutation, we use binary and real-valued operations as our chromosomes include both types of

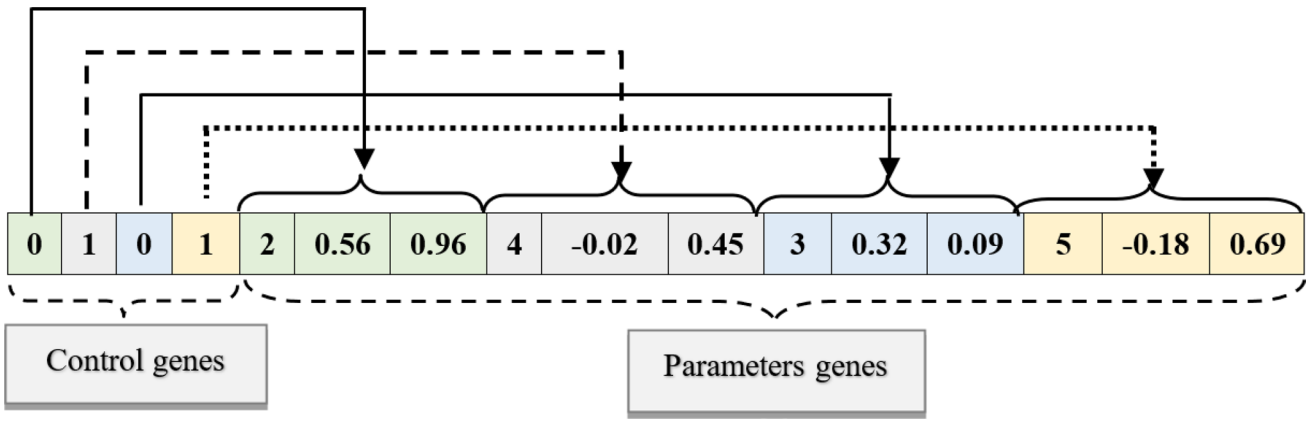


Fig. 6 An example of chromosome coding

numbers. Thus, the proposed algorithm can maintain more diversity in the population.

The training process can be summarized as follows:

(i) Stage 1: Defining RBFs The aim of this stage is to set the centers and the widths of RBFs.

- Step 0 (Input) : set the number of RBFs
- Step 1: Defining the centers of RBFs using K-means clustering
- Step 2: Calculating the widths of RBFs using the P-nearest neighbors' method

(ii) Stage 2: Adding neurons using GA The aim of this stage is to set the number, types, and parameters of the added neurons.

- Step 0 (Input): set the number of populations, GA parameters, Stopping criteria
- Step 1: Random initialisation of the input weights (W) and bias (b) Random Initialization of the genetic population
- Step 2: GA operations: selection; crossover; mutation
- Step 3: Evaluation Decoding: Set the number, types, and parameters of the added neurons Calculation of the output weights (β) using the pseud inverse method Calculation of the objective function (Eq. 13)
- Step 4: if Stopping criteria is satisfied (max iteration or objective function) then stop, if not return to step 2 (of stage 2).

3.4 Synthetic classification problem

To analyze the effect of the number and types of the added neurons, we performed some experiments on the synthetic example of Fig. 7. In these experiments, we first fixed the number of the initial RBFs then we applied the process of adding neurons using GA. For each number of RBFs, i.e. 10, 20 and 30, we performed four tests. The obtained results are reported in Table 1. We note that the numbers and types of the added neurons are different for each test, but they gave the same results in some cases. This shows that the process of adding neurons permits obtaining a set of complementary neurons despite their number and types.

On the other hand, we performed experiments with an RBFNN and ELMNN with different numbers of hidden neurons Table 2. We note that the proposed system gave better results than these two neural networks and it achieved these results with a smaller number of neurons. For example, with 27 and 30 neurons, the proposed classifier gave 99,5% and

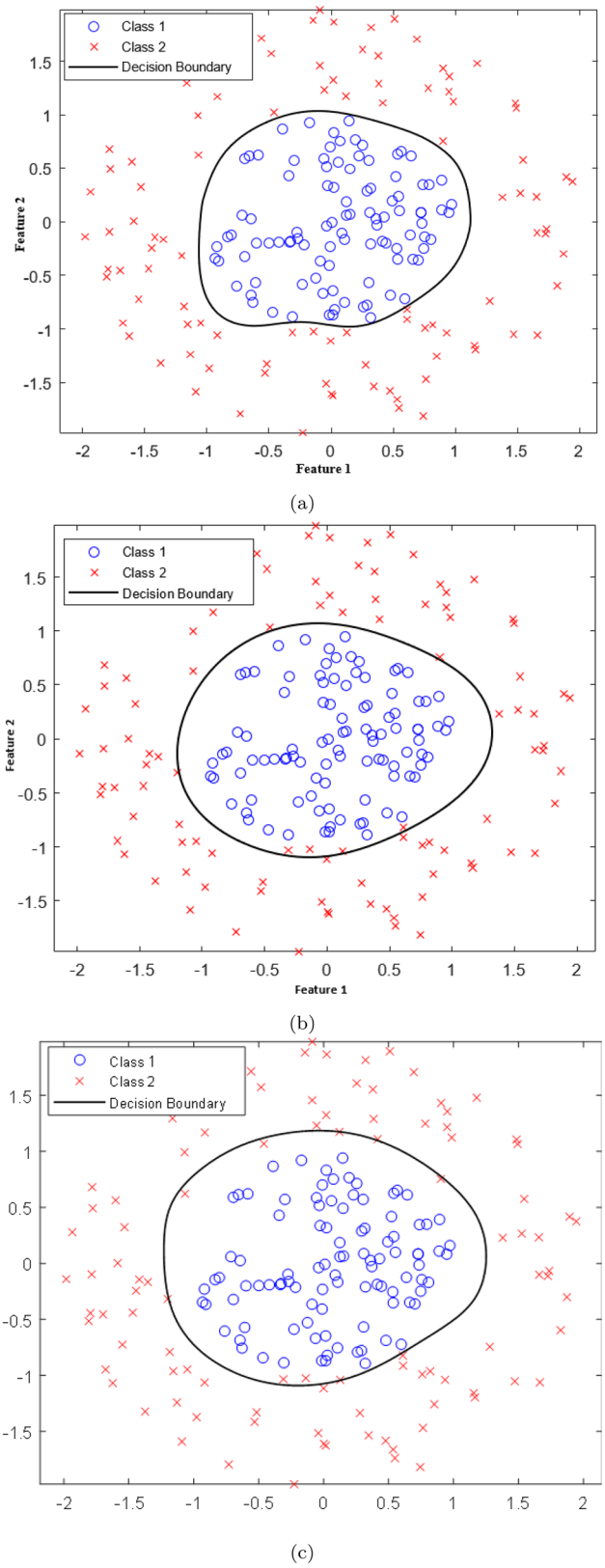


Fig. 7 A 2-D synthetic classification problem with two classes. **a** The decision boundary obtained using the proposed classifier with 27 neurons. **b** The decision boundary obtained using an RBFNN with 35 neurons. **c** The decision boundary obtained using an ELMNN with 35 neurons

Table 1 Results obtained using the proposed classifier with different numbers of RBFs and added neurons over the synthetic example

Model	#RBFs	Added neurons (Sigmoid, Gaussian, Hard-limit, Multi-quadratic)	#Total hidden neurons	Test accuracy (%)
Proposed classifier	10	(1/2/3/2) = 8	18	97.5
		(1/1/6/3) = 11	21	97.0
		(0/4/6/2) = 12	22	97.0
		(2/6/2/3) = 13	23	96.0
	20	(0/0/3/2) = 5	25	97.5
		(2/3/1/1) = 7	**27**	**99.55**
		(2/4/2/2) = 10	30	98.55
		(1/2/4/4) = 11	31	98.0
	30	(3/1/2/1) = 7	37	97.5
		(0/2/3/3) = 8	38	98.0
		(3/2/2/2) = 9	39	97.5
		(3/7/1/0) = 11	41	97.0

Bold values indicate the best results

Table 2 Results obtained using ELMNN and RBF with different numbers of hidden neurons over the synthetic example

Models	Type of activation functions	#Hidden neurons	Test accuracy (%)
ELMNN	Sigmoid	15	96.40 ± 0.49
		25	**96.80 ± 1.13**
		35	96.00 ± 1.02
		45	95.60 ± 0.80
RBFNN	Gaussian	15	97.25 ± 0.58
		25	96.80 ± 0.53
		35	97.20 ± 0.88
		45	**97.50 ± 0.57**

Bold values indicate the best results

98,55% respectively while the maximum accuracies given by RBFNN and ELMNN are, respectively, 97.5 % and 96.8 % even with 45 neurons.

Therefore, we conclude that it is possible to define a set of neurons with different types of the activation functions that can give better results than larger number of neurons with the same activation functions.

Table 1. The results obtained over the synthetic example of Fig. 7 using the proposed classifier with different numbers and types of the activation functions.

Table 2. The results obtained over the synthetic example of Fig. 7 using the RBFNN and ELMNN with different numbers of the hidden neurons.

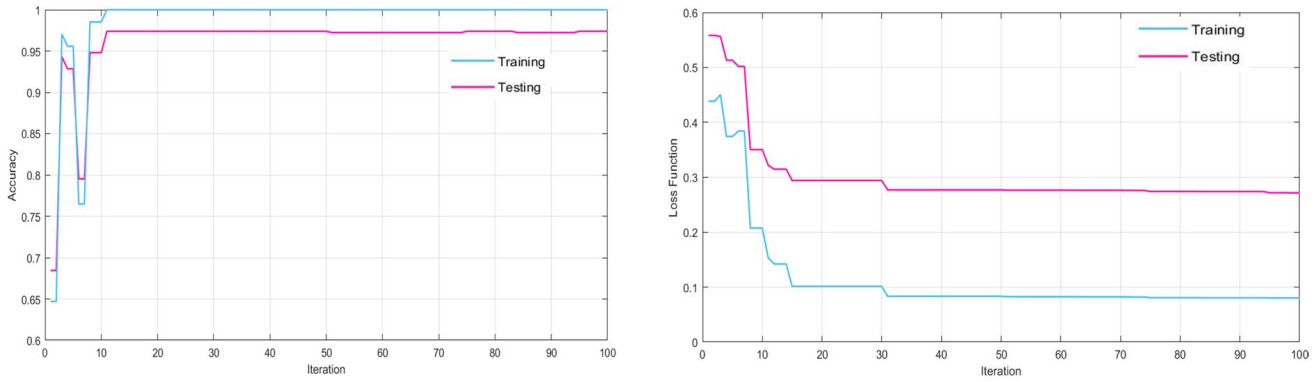
4 Tests and experiments

To assess the performance of the proposed classifier, we carried out tests on six medical datasets obtained from the UCI machine learning repository [39] and MIT-BIH arrhythmia dataset. These datasets are Wisconsin Breast Cancer, Pima Indians Diabetes, Heart-Statlog, Hepatitis, Parkinson and liver-disorders. In order to evaluate the generalization capabilities, we utilized a 10-fold cross validation. We opted for this method as it is the most commonly used in medical tests. Indeed, the higher value of K leads to a less biased model where significant variance might lead to over-fit, whereas the lower value of K is like the train-test split approach [40]. According to this strategy, each dataset is randomly divided into ten subsets: nine of them are used as training data, and the remaining subset is used as a test set. This procedure is repeated ten times and consequently, each sample appears once in a test set. Moreover, given the random initialization of the input weights and the initial population of the GA, we performed twenty runs for each test to make a fair evaluation of the proposed classifier.

For each dataset, we first performed tests with different numbers of RBFs to assess the effect of the different structures, and then we compared the proposed classifier with other work. The performance was measured with three evaluation methods, which are Accuracy, Specificity, and Sensitivity. The formulas for these parameters are given by:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \tag{16}$$

$$Specificity = \frac{TN}{TN + FP} \times 100\% \tag{17}$$



(a) Accuracy (b) Loss function

Fig. 8 Graphs of the performance of the proposed classifier on WBC dataset

Table 3 The confusion matrix obtained using the proposed method over the wisconsin breast cancer dataset

Classes	0	1	Total
0	**42**	2	44
1	0	**24**	24
Total	42	26	**68**

Bold values indicate the best results

$$Sensitivity = \frac{TP}{TP + FN} \times 100\% \tag{18}$$

where TP (True Positive) indicates the number of correct disease samples, FN (False Negative) indicates number of false healthy samples, TN (True Negative) indicates the number of correct healthy samples and FP (False Positive) indicates the number of false disease samples.

4.1 Wisconsin breast cancer (WBC) classification

This dataset was gathered by scientists from the University of Wisconsin Hospitals, Madison, USA. It contains 699 records, among them 16 are with missing data. To be coherent with the literature, we removed these records. The retained dataset consists then of 683 records, in which 239 examples are malignant and 444 are benign. Nine integer features characterize each record.

Figure 8 represents the convergence of the proposed classifier (accuracy and loss function graph) on the wisconsin breast cancer classification dataset.

The graph of the accuracy is shown in Fig. 8a, where the blue line corresponds to training data, obtaining a final accuracy of 1, and the pink line corresponds to the testing data, obtaining a final accuracy of 0.97.

The graph of the loss function is shown in Fig. 8b, where the blue line corresponds to the training data, obtaining a final value of 0.01, and the pink line corresponds to the testing data, obtaining a final value of 0.27.

Table 3 presents the confusion matrix obtained for the test data. This matrix represents the benign and malignant records in the WBC dataset. Each row represents the actual class, and each column represents the predicted class.

Table 4 displays the obtained results corresponding to different numbers of RBFs. The best results were obtained with 10 RBFs.

Table 5 compares the obtained results with those of some state-of-the-art works. These works include: neural networks, ELM, PSO, GA, Bee inspired, Naïve Bayesian, decision trees, clustering. PSO-RBFNN [43] and RBFNN-BBO [7] gave the best accuracy. Our system provided slightly lower accuracy, but with much fewer hidden neurons compared to PSO-RBFNN. Our classifier includes only 22 hidden neurons, while PSO-RBFNN includes 87. Moreover, in our work, we used cross validation while these two works used the hold-out method. Indeed, the cross validation gives

Table 4 Results obtained using the proposed classifier with different numbers of RBFs over WBC dataset

# RBFs	Accuracy (%)	Maximum	Minimum	Sensitivity (%)	Specificity (%)	# Added neurons
6	97.12 ± 0.33	97.80	96.65	97.99 ± 0.70	96.63 ± 0.31	13.16 ± 0.81
8	96.90 ± 0.29	97.36	96.48	97.94 ± 0.70	96.33 ± 0.40	12.90 ± 0.68
10	**97.38 ± 0.26**	**97.80**	**96.93**	**98.38 ± 0.45**	**96.85 ± 0.30**	**12.7 ± 0.73**
20	97.25 ± 0.24	97.66	96.92	98.28 ± 0.36	96.69 ± 0.30	12.29 ± 1.35

Bold values indicate the best results

Table 5 Comparison of the obtained results with previous works on WBC dataset

Authors	Methods	Comparison criterion				
		Accuracy(%)	Sensitivity (%)	Specificity (%)	# Hidden neurons	Cross validation
Carlos J. Mantas [41]	Credal C4.5	95.12	–	–	–	[10-CV]
Alisson S.C.A [15]	GAP-ELM	96.6 ± 1.7	–	–	13.7 ± 4.9	[10-CV]
Liangxiao Jiang [42]	DFW-NB	97.34 ± 1.02	–	–	–	[10-CV]
Dávila Patríciaz [4]	BeeRBF	96.87 ± 0.03	–	–	70.61 ± 1.48	[10-CV]
Cheruku Ramalingaswamy [43]	PSO-RBFNN	**97.86**	97.34	98.15	87	70–30%
Ibrahim Aljarah [7]	RBFNN-BBO	**97.86 ± 0.37**	98.09	97.41	10	67–33%
Md. Milon Islam [44]	SVM	97.14	100	92.3	–	[10–CV]
Rabab Bousmaha [45]	PLMVO-MLP	96.3 ± 0.001	–	–	–	66–34%
Proposed	RBFNN-ELM-GA	97.38 ± 0.26	98.38 ± 0.45	96.85 ± 0.3	10 +(12.7 ± 0.73)	[10-CV]

Bold values indicate the best results

DFW deep feature weighting, *NB* Naive Bayes, *SVM* support vector machine, *PLMVO* hybrid particle swarm optimization with Multi-verse optimization based on Lévy fight

better evaluation as the model is tested on multiple train-test splits and, consequently, on all data. On the other hand, the hold-out method depends on just one train-test split.

4.2 Pima Indians Diabetes (PID) classification

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. It contains 768 samples, in which 268 examples are diabetic and 500 are

Table 6 Results obtained using the proposed classifier with different numbers of RBFs over PID dataset

# RBFs	Accuracy (%)	Maximum	Minimum	Sensitivity (%)	Specificity (%)	# Added neurons
6	77.03 ± 0.58	77.90	76.20	58.43 ± 1.27	87.00 ± 0.74	7.48 ± 0.50
8	77.05 ± 0.46	77.99	76.68	57.67 ± 0.65	87.46 ± 0.49	7.13 ± 0.49
10	76.83 ± 0.58	77.89	76.34	56.95 ± 1.42	87.50 ± 0.58	7.21 ± 0.34
20	**77.61 ± 0.69**	**78.62**	**77.09**	**57.35 ± 1.24**	**88.22 ± 0.63**	**7.26 ± 0.45**
25	77.56 ± 0.49	78.23	77	57.25 ± 0.87	88.42 ± 0.52	7.06 ± 0.50

Bold values indicate the best results

Table 7 Comparison of the obtained results with previous works on PID dataset

Authors	Methods	Comparison criterion				
		Accuracy (%)	Sensitivity (%)	Specificity (%)	# Hidden neurons	Cross validation
Carlos J. Mantas [41]	Credal C4.5	74.15	–	–	–	[10-CV]
Zahra Beheshti [46]	CAPSO-MLP	72.99	85.14	69.27	–	80%-20%
Jenni Raitoharju [47]	RBF-CS-MDPSO	75.6 ± 0.9	–	–	–	[5-CV]
Alisson S.C.A [15]	GAP-ELM	74.4 ± 2.9	–	–	19.7 ± 4.0	[10-CV]
Dávila Patrícia [4]	BeeRBF	76.91 ± 0.57	–	–	35.54 ± 1.22	[10-CV]
Damodar Reddy Edla [48]	RBFNN-Bat-Opt	73.91	81.33	60.00	47	–
Cheruku Ramalingaswamy [43]	PSO-RBFNN	72.60	77.34	63.75	66	70%-30%
Cheruku R [5]	RBFNN-Ratio Index-Bat	70.00	77.34	56.25	43	70%-30%
Ömer Faruk. E [31]	RWNAFt	73.38	–	–	–	[5-CV]
Ibrahim Aljarah [7]	RBFNN-BBO	71.60 ± 1.9	46.25	86.27	10	67%-33%
Proposed	RBFNN-ELM-GA	**77.61 ± 0.69**	57.35 ± 1.24	88.22 ± 0.63	20+(7.26 ± 0.45)	[10-CV]

Bold values indicate the best results

not. In this dataset, there are no missing data points. Nine features characterize each sample.

Table 6 displays the obtained results corresponding to different numbers of RBFs. The best results were obtained with 20 RBFs.

Table 7 compares the obtained results with those of some state-of-the-art works. These works include a lot of techniques, such as MLP, ELM, PSO, GA, Bee inspired, Bat inspired, decision trees, clustering. In this database, our classifier gave the best accuracy. Concerning the number of hidden neurons, our classifier includes the minimum number compared to the other works except RBFNN-BBO [7].This classifier included only 10 hidden neurons but it was highly outperformed by the proposed classifier in term of accuracy.

4.3 Heart-Statlog classification

The data was obtained from the Cleveland Clinic Foundation. This dataset contains 270 observations and 2 classes: presence and absence of heart disease. In this dataset, there are no missing data. Each observation is characterized by thirteen features.

Table 8 displays the obtained results corresponding to different numbers of RBFs. With up of 25 hidden neurons, the proposed classifier gave good results.

Table 9 compares the obtained results with those of some state-of-the-art works. These works include MLP, PSO, GA, Bee inspired, Naïve Bayesian, decision trees, clustering. We note that the GAFL-System [49] gave the best accuracy. Among the compared works, only two mentioned the number of hidden neurons. Our classifier included much fewer than them.

Table 8 Results obtained using the proposed classifier with different numbers of RBFs over Heart-Statlog dataset

# RBFs	Accuracy (%)	Maximum	Minimum	Sensitivity (%)	Specificity (%)	# Added neurons
6	82.70 ± 0.99	83.70	80.74	77.67 ± 1.56	86.73 ± 1.42	4.60 ± 0.51
10	83.04 ± 1.21	85.13	80.92	79.17 ± 1.67	86.13 ± 1.36	4.63 ± 0.35
20	83.15 ± 0.55	84.98	81.11	77.85 ± 1.54	87.35 ± 0.96	4.28 ± 0.30
25	**83.71 ± 0.99**	**85.56**	**82.60**	**77.92 ± 1.64**	**88.34 ± 1.23**	**3.97 ± 0.31**
30	83.22 ± 0.74	84.44	81.85	77.75 ± .47	87.60 ± 1.0	3.93 ± 0.69

Bold values indicate the best results

Table 9 Comparison of the obtained results with previous works on Heart-Statlog dataset

Authors	Methods	Comparison criterion				
		Accuracy(%)	Sensitivity (%)	Specificity (%)	# Hidden neurons	Cross validation
Carlos J. Mantas [41]	Credal C4.5	80.33	–	–	–	[10-CV]
Zahra Beheshti [46]	CAPSO-MLP	81.85	74.63	90.21	–	80%-20%
T. Santhanam [49]	GAFL-System	**86**	80	90	–	[10-CV]
Liangxiao Jiang [42]	DFW-NB	83.63 ± 3.24	–	–	–	[10-CV]
Dávila Patrícia[4]	BeeRBF	59.12 ± 1.6	–	–	40.02 ± 1.076	[10-CV]
Cheruku Ramalingaswamy [43]	PSO-RBFNN	**85.76**	84.28	88.12	56	70%-30%
Rabab Bousmaha [45]	PLMVO-MLP	75.7 ± 0.063	–	–	–	66%-34%
Proposed	RBFNN-ELM-GA	83.71 ± 0.99	77.92 ± 1.64	88.34 ± 1.23	25+(3.97 ± 0.31)	[10-CV]

Bold values indicate the best results

Table 10 Results obtained using the proposed classifier with different numbers of RBFs over Hepatitis dataset

# RBFs	Accuracy (%)	Maximum	Minimum	Sensitivity (%)	Specificity (%)	# Added neurons
4	86.50 ± 1.65	88.75	83.75	96.07 ± 1.36	39.50 ± 6.85	4.31 ± 0.36
6	**87.10 ± 1.34**	**88.75**	**83.78**	**95.89 ± 1.35**	**40.10 ± 7.62**	**4.38 ± 0.51**
8	86 ± 1.66	87.50	83.23	95.72 ± 1.60	38 ± 9.18	4.30 ± 0.30
10	85.77 ± 1.48	87.50	82.32	94.86 ± 0.96	38 ± 9.50	4.20 ± 0.25
20	86.25 ± 1.77	88.75	82.50	94.60 ± 1.08	42 ± 8.23	4.22 ± 0.30

Bold values indicate the best results

Table 11 Comparison of the obtained results with previous works on Hepatitis dataset

Authors	Methods	Comparison criterion				
		Accuracy(%)	Sensitivity (%)	Specificity (%)	# Hidden neurons	Cross validation
Carlos J. Mantas [41]	Credal C4.5	79.79	–	–	–	[10-CV]
Zahra Beheshti [46]	CAPSO-MLP	71.29	72.82	65.33	–	80%-20%
Liangxiao Jiang [42]	DFW-NB	83.87 ± 5.89	–	–	–	[10-CV]
Ibrahim Aljarah [7]	RBFNN-BBO	84.72 ± 1.65	59.00	90.70	6	67%-33%
Proposed	RBFNN-ELM-GA	**87.10 ± 1.34**	95.89 ± 1.35	40.10 ± 7.62	6+(4.38 ± 0.5)	[10-CV]

Bold values indicate the best results

Table 12 Results obtained using the proposed classifier with different numbers of RBFs over Parkinson disease dataset

# RBFs	Accuracy (%)	Maximum	Minimum	Sensitivity (%)	Specificity (%)	# Added neurons
6	87.06 ± 1.45	88.68	84.13	95.82 ± 1.15	60.45 ± 4.72	7.68 ± 1.06
8	88.63 ± 1.74	91.26	86.18	95.61 ± 1.36	67.10 ± 5.62	6.22 ± 0.48
10	89.99 ± 1.50	92.80	88.25	96.76 ± 1.33	69.15 ± 3.77	5.79 ± 0.67
20	**92.62 ± 1.20**	**93.98**	**90.88**	**96.50 ± 1.10**	**80.76 ± 5.45**	**4.96 ± 0.55**
25	92.01 ± 1.08	94.38	90.28	96.38 ± 1.32	78.45 ± 5.19	4.78 ± 0.52

Bold values indicate the best results

Table 13 Comparison of the obtained results with previous works on Parkinson disease dataset

Authors	Methods	Comparison criterion				
		Accuracy(%)	Sensitivity (%)	Specificity (%)	# Hidden neurons	Cross validation
Zahra Beheshti [46]	CAPSO-MLP	92.56	93.96	85.48	–	80%-20%
Jenni Raitoharju [47]	RBF-CS-MDPSO	85.7 ± 1.20	–	–	–	[5-CV]
Ibrahim Aljarah [7]	RBFNN-BBO	86.42 ± 1.92	99.61	44.37	10	67%-33%
Proposed	RBFNN-ELM-GA	**92.62 ± 1.20**	96.50 ± 1.10	80.76 ± 5.45	20+(4.96 ± 0.5)	[10-CV]

Bold values indicate the best results

4.4 Hepatitis classification

The dataset is obtained from the Carnegie-Mellon University. It contains 155 instances belonging to two classes: live or die. Each instance is characterized by 19 features.

Table 10 displays the results obtained with different numbers of RBFs. We note that the proposed classifier gave good results with 6 RBFs.

Table 11 compares the obtained results with those of some state-of-the-art works. These works include neural networks, MLP, PSO, GA, Naïve Bayesian, decision trees, clustering. We note that our classifier gave the best accuracy. Concerning the number of hidden neurons, it was mentioned only in RBFNN-BBO [7]. In this work, only six hidden neurons were used, but our classifier provided much higher accuracy.

4.5 Parkinson disease classification

This dataset was created at the University of Oxford in collaboration with the National Centre for Voice and Speech, Denver, Colorado. This dataset contains 195 records and

Table 14 Results obtained using the proposed classifier with different numbers of RBFs over BUPA-Liver disorder dataset

# RBFs	Accuracy (%)	Maximum	Minimum	Sensitivity (%)	Specificity (%)	# Added neurons
6	71.09 ± 1.22	73.35	70.10	81.10 ± 2.04	57.28 ± 1.98	7.58 ± 0.32
8	70.84 ± 1.96	72.88	68.92	80.50 ± 2.20	57.55 ± 2.42	7.39 ± 0.42
10	**72.48 ± 1.02**	**73.98**	**70.20**	**82.68 ± 1.08**	**58.39 ± 2.15**	**7.01 ± 0.48**
20	72.28 ± 1.05	73.98	71.26	81.70 ± 1.55	59.36 ± 1.92	7.46 ± 0.43

Bold values indicate the best results

Table 15 Comparison of the obtained results with previous works on BUPA-Liver disorder dataset

Authors	Methods	Comparison criterion				
		Accuracy(%)	Sensitivity (%)	Specificity (%)	# Hidden neurons	Cross validation
Carlos J. Mantas [41]	Credal C4.5	64.53	–	–	–	[10-CV]
Zahra Beheshti [46]	CAPSO-MLP	72.32	76.57	65.49	–	80%-20%
Ömer Faruk. E [31]	RWNAFt	70.43	–	–	–	[5-CV]
Ibrahim Aljarah [7]	RBFNN-BBO	69.15 ± 4.3	43.20	88.24	10	67%-33%
Proposed	RBFNN-ELM-GA	**72.48 ± 1.02**	82.68 ± 1.08	58.39 ± 2.15	10+(7.01 ± 0.48)	[10-CV]

Bold values indicate the best results

CAPSO centripetal accelerated particle swarm optimization, *RWNAFt* random weight artificial neural network with trained activation function

2 classes: the presence or absence of Parkinson disease. Twenty-two features characterize each record.

Table 12 displays the obtained results with different numbers of RBFs. The best performances were obtained with 20 RBFs and with up to 20 neurons, the accuracy decreased.

Table 13 compares the obtained results with those of other works. These works include neural networks, MLP, PSO, clustering. Our classifier gave the best results. Concerning the number of hidden neurons, RBFNN-BBO [7] included less number, but with lower accuracy compared to our classifier.

4.6 BUPA liver disorders classification

This dataset was created at BUPA Medical Research Ltd. It consists of 345 samples and 2 classes. 200 samples belong to the first class and 145 samples to the second.

Table 14 displays the obtained results with different numbers of RBFs. The best results were obtained with 10 RBFs. With up to 20 neurons, the accuracy decreased slightly.

Table 15 compares the obtained results with those of other works. These works include neural networks, MLP, PSO, Naïve Bayesian, decision trees, clustering. Our classifier gave the best results. Concerning the number of hidden neurons, RBFNN-BBO [7] included less number, but our classifier outperformed it. Indeed, this classifier included 10 hidden neurons and gave an average accuracy of 69.15%, while our classifier included 10 RBFs plus 7 added neurons and gave an average accuracy of 72.48%. We note that our accuracy is remarkably higher.

4.7 MIT-BIH arrhythmia dataset (inter-patient paradigm)

To evaluate the performance of the proposed classifier, we performed tests on the PhysioNet MIT-BIH Arrhythmia dataset [50]. This dataset involves ECG signals collected at a sampling rate of 360 Hz for 48 distinct patients.

In this study, the inter-patient paradigm was considered. The training and test sets are constructed from different patients in this paradigm. In this work, we adopt the protocol proposed by de Chazal et al. [51], which is widely adopted in the literature. The heartbeats from the MIT-BIH dataset (44 records according to AAMI) are divided into two sets of records according to this protocol: Dataset1 = 101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 219, 212, 214, 219, 221, 222, 228, 231,

Table 16 Comparison of the obtained results with previous works on MIT-BIH arrhythmia dataset (inter-patient paradigm)

Authors	Methods	Comparison criterion		
		Accuracy(%)	Sensitivity (%)	Specificity (%)
WANG Tao [52]	CNN + RR-intervals + MLP	92.53	–	–
WANG Haoren [53]	CNN	93.4	–	–
Siouda Roguia [54]	SSAEs + MLPs	**94.69**	71	89.53
YAN Zhanglu [55]	Raw heartbeat data + CNN + SNN	90	–	–
Houssein Essam [56]	RR, Wavelet, LBP, HOS, morphological features + SVM	**98.26**	97.43	–
Siouda Roguia [57]	SAE +RR-intervals + RBFNN + RVFLN	93.11	55.66	80.55
This study	RBFNN+ELM+GA	93.86	68.15	99.21

CNN convolutional neural network, *SSAE* stacked sparse autoencoders, *SNN* spiking neural network, *LBP* local binary pattern, *HOS* higher-order statistical, *RVFLN* random vector functional link network

232, Dataset1 is used to train the classification systems and Dataset2 is used for testing. In this separation, there is no concern about including the heartbeats of the same patient in the training and test sets.

Table 16 compares the obtained results with other works applied to the same dataset, i.e., MIT-BIH arrhythmia. These works include a large variety of classification methods and feature extracting techniques.

From Table 16 , we notice that the performance of our classifier outperformed all these works, except two works. Although these works provided better results, our system is much simpler. The first work, Siouda et al. [54], uses deep-stacked autoencoders and a system of multiple MLPs. The second work, Houssein et al. [56], uses a lot of features, i.e. RR intervals, wavelets, local binary patterns, higher-order statistics, and other morphological features.

5 Conclusion and future work

This paper introduced a neural classifier for medical diagnosis based on combining RBF and ELM neural networks. The aim is to make use of the advantages and the complementary properties of these two types of neural networks. The basic idea relies on using an RBFNN with minimal structure complemented by an ELMNN that contains a diversity of hidden neurons. The optimization of the activation functions in the ELM is performed using a genetic algorithm.

The proposed system was first tested over a synthetic classification problem. We noted that the process of adding neurons, in the proposed classifier, permitted obtaining a set of complementary neurons despite their number and types. We also noted that it gave better results than RBFNN and ELMNN and that it required a smaller number of neurons. The proposed classifier was then tested on six medical datasets from the UCI machine-learning repository and also MIT-BIH arrhythmia dataset. These datasets include a large variety of features. The proposed classifier was compared with some state-of-the-art methods in terms of accuracy and number of hidden neurons. The obtained results are promising as the proposed classifier provided either better results and/or a smaller number of hidden neurons in most comparisons.

The approach presented in this paper constitutes a general methodology that consists in complementing a compact RBFNN with an ELMNN that contains a diversity of activation functions. In this work, we used K-means and P-nearest neighbors' methods for defining RBFs and used GA for optimizing the added neurons. Besides, we employed four types of activation functions. Therefore, to have better performances, other variants can be considered. This can be done by: (i) using other methods for defining the RBFs; (ii) considering other metaheuristics for optimizing the added neurons; (iii) using other types of activation functions.

As future work, we plan to further explore the idea of combining/complementing prior knowledge-based classifiers, like RBF, with diverse activation functions based-neural networks. This would considerably reduce the execution time of the optimization process because it concerns only the added neurons. This could also permit an automatic setting of compact models. In fact, deep neural networks are time-consuming, and our aim is to set shallow, or at least less deep, neural models. This way, the proposed models can be easily integrated in applications requiring fast, simple systems like remote health supervision, wireless body area networks, and medical wearable devices.

In addition, we think that interpretability can be enhanced by the proposed approach because the obtained model is essentially based on the prior knowledge based-classifier. This way, we plan to introduce fully interpretable networks, like the model presented in [58], or networks that permit extracting knowledge in the form of fuzzy if then rules, like the models presented in [59, 60].

References

1. Broomhead DS, Lowe D (1988) Radial basis functions, multivariable functional interpolation and adaptive networks. Royal Signals and Radar Establishment Malvern (United Kingdom)
2. Huang W, Oh SK, Pedrycz W (2014) Design of hybrid radial basis function neural networks (HRBFNNs) realized with the aid of hybridization of fuzzy clustering method (FCM) and polynomial neural networks (PNNs). Neural Netw 60:166–181
3. Alexandridis A, Chondrodima E, Sarimveis H (2016) Cooperative learning for radial basis function networks using particle swarm optimization. Appl Soft Comput 49:485–497
4. Cruz DPF, Maia RD, da Silva LA, de Castro LN (2016) BeeRBF: a bee-inspired data clustering approach to design RBF neural network classifiers. Neurocomputing 172:427–437
5. Cheruku R, Edla DR, Kuppili V (2017) Diabetes classification using radial basis function network by combining cluster validity index and bat optimization with novel fitness function. Int J Comput Intell Syst 10(1):247–265
6. Hu Y, You JJ, Liu JN, He T (2018) An eigenvector based center selection for fast training scheme of RBFNN. Inf Sci 428:62–75
7. Aljarah I, Faris H, Mirjalili S, Al-Madi N (2018) Training radial basis function networks using biogeography-based optimizer. Neural Comput Appl 29(7):529–553
8. Dey P, Gopal M, Pradhan P, Pal T (2019) On robustness of radial basis function network with input perturbation. Neural Comput Appl 31(2):523–537
9. Roguia S, Mohamed N (2019) An optimized RBF-neural network for breast cancer classification. Int J Inform Appl Math 1(1):24–34
10. Siouda R, Nemissi M, Seridi H (2020) A genetic algorithm-based deep RBF neural network for medical classification. In: Proceedings of the 1st international conference on intelligent systems and pattern recognition, pp 27–32
11. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
12. Chen X, Xie W, Yu S (2020) Body fat percentage prediction algorithm based on PSO-ELM model and BIA. In Proceedings

of 2020 the 6th international conference on computing and data engineering, pp 5–8
13. Tian Z, Ren Y, Wang G (2019) Short-term wind speed prediction based on improved PSO algorithm optimized EM-ELM. Energy Sources Part A: Recovery Util Environ Eff 41(1):26–46
14. Nemissi M, Salah H, Seridi H (2018) Breast cancer diagnosis using an enhanced extreme learning machine based-neural network. In 2018 international conference on signal, image, vision and their applications (SIVA), pp 1–4. IEEE
15. Alencar AS, Neto ARR, Gomes JPP (2016) A new pruning method for extreme learning machines via genetic algorithms. Appl Soft Comput 44:101–107
16. Mohapatra P, Chakravarty S, Dash PK (2015) An improved cuckoo search based extreme learning machine for medical data classification. Swarm Evol Comput 24:25–49
17. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recogn 38(10):1759–1763
18. Huang GB, Siew CK (2004) Extreme learning machine: RBF network case. In: ICARCV 2004 8th control, automation, robotics and vision conference, 2004, vol 2, pp 1029–1036. IEEE
19. Xu X, Tian S (2016) ELM-RBF neural networks using microgenetic algorithm for optimization. Int J Hybrid Inf Technol 9(12):27–36
20. Wu Y, Chen Z, Wu L, Lin P, Cheng S, Lu P (2017) An intelligent fault diagnosis approach for PV array based on SA-RBF kernel extreme learning machine. Energy Procedia 105:1070–1076
21. Xu X, Shan D, Li S, Sun T, Xiao P, Fan J (2019) Multi-label learning method based on ML-RBF and Laplacian ELM. Neurocomputing 331:213–219
22. Wen H, Fan H, Xie W, Pei J (2017) Hybrid structure-adaptive RBF-ELM network classifier. IEEE Access 5:16539–16554
23. Xia L, Hu P, Ma K, Yang L (2021) Research on measurement modeling of spherical joint rotation angle based on RBF-ELM network. IEEE Sens J 21(20):23118–23124
24. Qasem SN, Shamsuddin SM (2010) Generalization improvement of radial basis function network based on multi-objective particle swarm optimization. J Artif Intell 3(1):1–16
25. Deng W, Zheng Q, Chen L (2009) Regularized extreme learning machine. In: 2009 IEEE symposium on computational intelligence and data mining, pp 389–395. IEEE
26. Garcia-Capulin CH, Cuevas FJ, Trejo-Caballero G, Rostro-Gonzalez H (2015) A hierarchical genetic algorithm approach for curve fitting with B-splines. Genet Program Evolvable Mach 16(2):151–166
27. Melin P, Sánchez D (2019) Optimization of type-1, interval type-2 and general type-2 fuzzy inference systems using a hierarchical genetic algorithm for modular granular neural networks. Granul Comput 4(2):211–236
28. Zhao G, Shen Z, Man Z (2011) Robust input weight selection for well-conditioned extreme learning machine. Int J Inf Technol 17(1):1–13
29. Han F, Yao HF, Ling QH (2013) An improved evolutionary extreme learning machine based on particle swarm optimization. Neurocomputing 116:87–93
30. Li B, Li Y, Rong X (2013) The extreme learning machine learning algorithm with tunable activation function. Neural Comput Appl 22(3):531–539
31. Ertuğrul ÖF (2018) A novel type of activation function in artificial neural networks: trained activation function. Neural Netw 99:148–157
32. López-Rubio E, Ortega-Zamorano F, Domínguez E, Muñoz-Pérez J (2019) Piecewise polynomial activation functions for feedforward neural networks. Neural Process Lett 50(1):121–147
33. Farhadi F, Nia VP, Lodi A (2019) Activation adaptation in neural networks. arXiv preprint arXiv:1901.09849
34. Qian S, Liu H, Liu C, Wu S, San Wong H (2018) Adaptive activation functions in convolutional neural networks. Neurocomputing 272:204–212
35. Huang GB (2014) An insight into extreme learning machines: random neurons, random features and kernels. Cogn Comput 6(3):376–390
36. Huang GB, Zhou H, Ding X, Zhang R (2011) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B (Cybern) 42(2):513–529
37. Nemissi M, Seridi H, Akdag H (2014) One-against-all and one-against-one based neuro-fuzzy classifiers. J Intell Fuzzy Syst 26(6):2661–2670
38. Benoudjit N, Archambeau C, Lendasse A, Lee JA, Verleysen M (2002) Width optimization of the Gaussian kernels in radial basis function networks. In: ESANN, vol 2, pp 425–432
39. Bache K, Lichman M (2013) UCI machine learning repository
40. Rafało M (2021) Cross validation methods: analysis based on diagnostics of thyroid cancer metastasis. ICT Express
41. Mantas CJ, Abellan J (2014) Credal-C4. 5: decision tree based on imprecise probabilities to classify noisy data. Expert Syst Appl 41(10):4625–4637
42. Jiang L, Li C, Wang S, Zhang L (2016) Deep feature weighting for Naive Bayes and its application to text classification. Eng Appl Artif Intell 52:26–39
43. Cheruku R, Edla DR, Kuppili V, Dharavath R (2017) PSO-RBFNN: a PSO-based clustering approach for RBFNN design to classify disease data. In: International conference on artificial neural networks, pp 411–419. Springer, Cham
44. Islam MM, Haque MR, Iqbal H, Hasan MM, Hasan M, Kabir MN (2020) Breast cancer prediction: a comparative study using machine learning techniques. SN Comput Sci 1(5):1–14
45. Bousmaha R, Hamou RM, Amine A (2021) Automatic selection of hidden neurons and weights in neural networks for data classification using hybrid particle swarm optimization, multi-verse optimization based on Lévy flight. Evolut Intell, 1–20
46. Beheshti Z, Shamsuddin SMH, Beheshti E, Yuhaniz SS (2014) Enhancement of artificial neural network learning using centripetal accelerated particle swarm optimization for medical diseases diagnosis. Soft Comput 18(11):2253–2270
47. Raitoharju J, Kiranyaz S, Gabbouj M (2015) Training radial basis function neural networks for classification via class-specific clustering. IEEE Trans Neural Netw Learn Syst 27(12):2458–2471
48. Edla DR, Cheruku R (2017) Diabetes-finder: a bat optimized classification system for type-2 diabetes. Procedia Comput Sci 115:235–242
49. Santhanam T, Ephzibah EP (2015) Heart disease prediction using hybrid genetic fuzzy model. Indian J Sci Technol 8(9):797
50. PhysioNet (2001) PhysioNet: MIT-BIH arrhythmia database. Phys-ioNet: MIT-BIH arrhythmia database. https://archive.physionet.org/cgi-bin/atm/ATM. Accessed 30 Jan 2022
51. De Chazal P, O'Dwyer M, Reilly RB (2004) Automatic classification of heartbeats using ECG morphology and heartbeat interval features. IEEE Trans Biomed Eng 51(7):1196–1206
52. Wang T, Lu C, Yang M, Hong F, Liu C (2020) A hybrid method for heartbeat classification via convolutional neural networks, multilayer perceptrons and focal loss. PeerJ Comput Sci 6:e324
53. Wang H, Shi H, Lin K, Qin C, Zhao L, Huang Y, Liu C (2020) A high-precision arrhythmia classification method based on dual fully connected neural network. Biomed Signal Process Control 58:101874
54. Siouda R, Nemissi M, Seridi H (2021) ECG beat classification using neural classifier based on deep autoencoder and decomposition techniques. Prog Artif Intell 10(3):333–347
55. Yan Z, Zhou J, Wong WF (2021) Energy efficient ECG classification with spiking neural network. Biomed Signal Process Control 63:102170

56. Houssein EH, Ibrahim IE, Neggaz N, Hassaballah M, Wazery YM (2021) An efficient ECG arrhythmia classification method based on Manta ray foraging optimization. Expert Syst Appl 181:115131
57. Siouda R, Nemissi M, Seridi H (2022) A random deep neural system for heartbeat classification. Evolv Syst, 1–12
58. Wang D, Chen Y, Shen C, Zhong J, Peng Z, Li C (2022) Fully interpretable neural network for locating resonance frequency bands for machine condition monitoring. Mech Syst Signal Process 168:108673
59. Colace F, Loia V, Tomasiello S (2019) Revising recurrent neural networks from a granular perspective. Appl Soft Comput 82:105535
60. Tomasiello S, Loia V, Khaliq A (2021) A granular recurrent neural network for multiple time series prediction. Neural Comput Appl 33(16):10293–10310