International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 9 (2019), pp. 1518-1523
© International Research Publication House. http://www.irphouse.com

Mechanism of automatic re-configuration of artificial neural networks for the identification of patterns

Holman Montiel Ariza[1], Fredy H. Martínez S.[2] and Edwar Jacinto Gómez[3]

[1, 2,3] *Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá D.C, Colombia.*

[1]*ORCID:* 0000-0002-6077-3510 [2]*ORCID*: 0000-0002-7258-3909, [3]*ORCID: 0000-0003-4038-8137*

Abstract

Artificial neural networks are bio-inspired mechanisms that try to reproduce exactly a set of data given by a user. Normally, the data set is used to train the network using a technique that modifies the configuration parameters of each neuron. However, finding the appropriate parameters for a neural network to classify or perform a regression that represents the given set is not a simple task, due to the large number of possible configurations and topologies that it may have. This paper proposes an adaptive neural network model that adjusts its configuration and topology automatically, to reduce the systematic error that a user induces when creating a network manually. This model is based on a multi-objective optimization technique (*G*enetic Algorithm), which minimizes computational resources and finds the necessary parameters for the neural network to work properly, even without defining the data types of the training set.

Keywords: Deep Learning, Genetic Algorithm, Artificial Neural Networks, Optimization.

1. INTRODUCTION

Artificial Neural Networks (ANNs) are bio-inspired mathematical models that emulate the functioning of the human brain by recreating the synapse phenomenon between neurons [1]. Normally, these types of models are used to perform classification or regression tasks. Classification tasks consist in the grouping of data according to a characteristic, for example, the identification or correction of characteristics in an image [2-4]. Regression tasks are performed to find approximate functions to groups of numerical data, for example, the reconstruction of polynomials or signals [5, 6].

ANNs establish many relationships to represent input and output information. These relationships are mathematical expressions depending on the type of neural networks and its components that are estimated using a training algorithm. The most common training method is back-propagation, which implements a descendant gradient strategy to reduce the mean square error and thus estimate the thresholds and weights of each neuron in the network. These neurons are grouped to fulfill specific functions of the network and are known as groups of neurons or layers, whose activation depends on a mathematical function that emulates the action potential of a real neuron [7, 8].

The combination between groups of neurons (topology) and the definition of their parameters (configuration) such as the activation function, the training method, the number of neurons and layers, determine network characteristics like the speed of convergence, the accuracy of the output or the margin of error [8]. Although, these characteristics allow an ANN to function in an adaptive way, they do not guarantee an appropriate functioning when the set of samples to be classified has redundancy, large dimensions or irrelevant characteristics that reduce the accuracy and increase the complexity of the classification. [7]

Some techniques to increase the accuracy of the output of a neural network have been previously proposed [9-11]. However, it is a subject that is under study, since the margin of error of an ANN depends on the training speed and learning rate, which vary depending on fixed parameters of the network (number of layers or neurons). In other words, when the topology of the network decreases the training time it is possible that the solution found is not suitable for the problem to be solved [7].

Considering the subject under study, this paper proposes a strategy to automate the process of ANN generation using a multi-objective optimization algorithm, which determines the appropriate configuration parameters for the ANN to solve a problem. The characteristics of the optimization algorithm and the ANN are described in detail in the following sections that are organized as follows: Section two (2) contains a description of the type of ANN and the optimization algorithm. Section three (3) describes the experiment used to perform network tests. In section four (4) the results obtained are found. In section five (5) there are the conclusions apprehended by developing this work.

2. MATERIALS AND METHODS

In relation to the previous section, machine or automatic learning is a compendium of techniques that are responsible for making representations of groups of data provided by a user. Currently, these techniques have been improved with strategies such as deep learning, which allow the user to make representations of information more complex than those made with traditional learning techniques. However, deep learning is an area under study and its algorithms are designed to perform cloud computing, due to the large amount of computational resources necessary for the implemented technique to work properly. In this paper there are presented two learning techniques that attempt to improve the limitation of deep learning networks, by automating the way to generate a neural network through a multi-objective optimization algorithm. These two learning techniques are described below [13,14].

2.1 Neural networks.

Neural networks are approximate models that allow to predict data sets and between them there are several classes depending on the type of work data. Normally a neural network has an input layer, an output layer and in some cases several intermediate or hidden layers. The input layer is the one that relates the input variables or variations of the magnitude that modify the output value. The output layer only shows the behavior of the network that is built when modifying the input value. The hidden layers are composed of auxiliary neurons that allow to increase the accuracy of the network, by implementing more coefficients in the approximate model [1-3].

The layers of the network are composed of neurons that store a coefficient or weight, which estimates an output value from a given input value (except class classifiers, since they do not have a defined input). The coefficients of the neurons are updated by training algorithms (for example: forward or backward propagation algorithms), which are responsible for reducing the margin of error between the training data and the data provided by the approximate model of the network (to estimate the error, functions such as the square error or square logarithmic error are implemented) [1].

Another characteristic of neural networks is that the output value is limited by a function, whose function is to prevent a value from exceeding a threshold before spreading to another neuron (some of the most common activation functions are: sigmoid, hyperbolic tangent or exponential). Considering that each initial neuron is assigned an initial weight before starting its training (they can be zeros, ones or random numbers) [1].

A deep learning neural network (DNN) is based on a conventional network model, with the advantage that it allows establishing linear relationships or not between the input and output variables. In addition, DNNs can have many layers which allows you to decompose a group of base data into many variables, to identify trends or patterns and thus predict an output value.

However, the output value depends on the processing of the input data, since to train a DNN it must be considered that the data type of the training set must be the same. That is, if the data set is made up of numbers and classes, the numbers must be converted to classes or assign a numerical value to each class (pre-processing).

In this paper the proposed model can encode the data in classes or numbers, depending on the level of accuracy that is achieved in each case. In addition, a DNN was implemented that reconfigures its characteristics according to the execution time and performance when classifying a given data set. In Fig. 1 a basic scheme of the modified DNN is shown and its main characteristics are described in the following numeral.

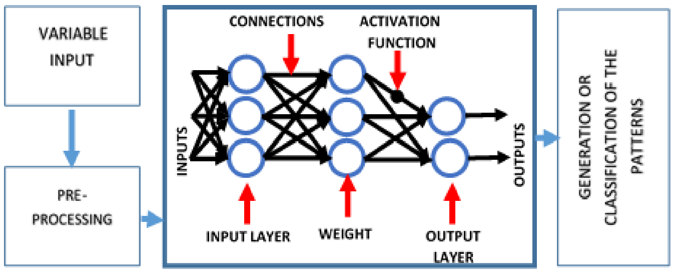


Fig. 1. Diagram of the operation of the DNN.

2.2 Algorithm of multi-objective optimization.

Optimization algorithms are normally used to determine a maximum or minimum value of a function (local or global optimum). There are many optimization techniques and among them there are genetic algorithms, which are bio-inspired algorithms based on Darwinian natural selection. This type of algorithm is population-based, that is, it optimizes a function from sets of solutions. Each component of the set is known as an individual and the value assigned to it when trying to solve the function is called the aptitude value, which is made up of one (mono-objective) or more values (multi-objective) [7].

The re-configurable neural network model proposed in this paper is based on a multi-objective genetic algorithm that adjusts the configuration, topology and parameters of a neural network, based on its performance and execution time. Initially, the proposed strategy encodes seven (7) main characteristics of a neural network in an array to create an individual. These characteristics are the number of neurons (β) and hidden layers (α), the algorithm for assigning initial values to weights of neurons (δ), activation functions (τ), optimization of weights (ω) and calculation of the output error (φ) and a metric to measure the performance of the network (ε).

α and β are calculated with a random number generator ($0 < \alpha < 1000, 1 < \beta < 1000$) with uniform distribution and the other components (δ, τ, ω, φ, ε) are assigned randomly from the definitions contained in a library (see Table 1). That is, α and β are integers and δ, τ, ω, φ, ε are labels that designate a characteristic of the network.

Table 1. Function library used to create an individual.

$\boldsymbol{\delta}$	$\boldsymbol{\tau}$	$\boldsymbol{\omega}$	$\boldsymbol{\varphi}$	$\boldsymbol{\varepsilon}$
Random uniform.	Tanh.	SGD	Mean squared error.	Binary accuracy
Ones.	Sigmoid.	RMSprop	Absolute squared error.	Categorical accuracy
Zeros.	Hard sigmoid.	Adagrad	Mean absolute percentage squared error.	Sparse categorical accuracy
Lecun normal.	Exponential.	Adadelta	Mean squared logarithmic error.	
Glorot uniform.	Lineal.	Adam	Squared hinge.	
He normal.	Selu.	Adamax	Hinge.	
He uniform.	Softplus.	Nadam	Logcosh.	
	Softsign.		Poisson.	
	Relu.		Cosine proximity.	
			Categorical hinge.	

The aptitude value (f (x)) of everyone is composed of two values that are the execution time (t) and the performance of the network (r). The main objective of the optimization technique is to minimize t and maximize r; to relate these values a combination of linear factors was proposed as shown in Equation 1.

$$if\ x = [\alpha, \beta, \delta, \tau, \omega, \varphi, \varepsilon]: x \neq \infty \rightarrow f(x) = r - t \qquad (1)$$

By performing this combination f (x) becomes a mono-objective function, where r has valuesbetween zero (0 = worst value) and one (1 = best value) and t> 0. As it is observed, when increasing the value of t, the value of f (x) decreases. Therefore, the proposed model attempts to maximize the value of f (x) by decreasing the value of t and increasing the value of r. The optimization of aptitude value is subject to the evolutionary process of the genetic algorithm (see Algorithm 1), which is composed of four stages that are selection, recombination, mutation and generational replacement.

Algorithm 1. Re-configuration model of the neural network

```
    AG_DNN Program ()
1.      C←Define number of individuals.
2.      G←Define number of generations.
3.      K←Assign training data from the network.
4.      K← Unify data type (K).
5.      For i←1 to C
6.          P0[i] ←x
    /*** Start optimization algorithm ***/
7.      For i←1 to G
    /*** Evaluation of the aptitude function ***/
8.          For j←1 to C
9.              $Y$←Creating DNN(P0[j])
10.             $[P, t]$←Train $(Y, t)$
11.             $r$←Compare $(P, K)$
12.             $f(x)[j]$←$r - t$
    /*** Genetic operators ***/
13.         P1← Tournament $(4, f(x), P0)$
14.         Father←P1
15.         Mother←Copy(A)
16.         Children←Recombination(Father,Mother)
17.         Children←Mutation (Children)
18.         P0←Children
End AG_DNN
```

The first stage is a tournament of four (4) individuals that are selected at random. Then these four individuals are grouped into two (2) groups, from which the individuals with the best aptitude value are selected and from the two selected individuals the one with the best aptitude value is selected. This routine is repeated until a new population (P1) is completed with the same number of individuals as the original population (P0).

The second stage consists of a recombination of individuals of two (2) lists. This is done by cloning the population, then ordered to the original list (Father) from highest to lowest aptitude value and the cloned list (Mother) from least to greatest. Once ordered you generate a population of Children by combining characteristics of the parents. The third stage consists in taking an individual from the Children list and changing one of its components for another, in both cases the selection is carried out randomly.

Finally, the generational replacement is made by assigning Children to the original population. This process repeats several iterations (generations) established by the user and can be observed in detail in Fig. 2.

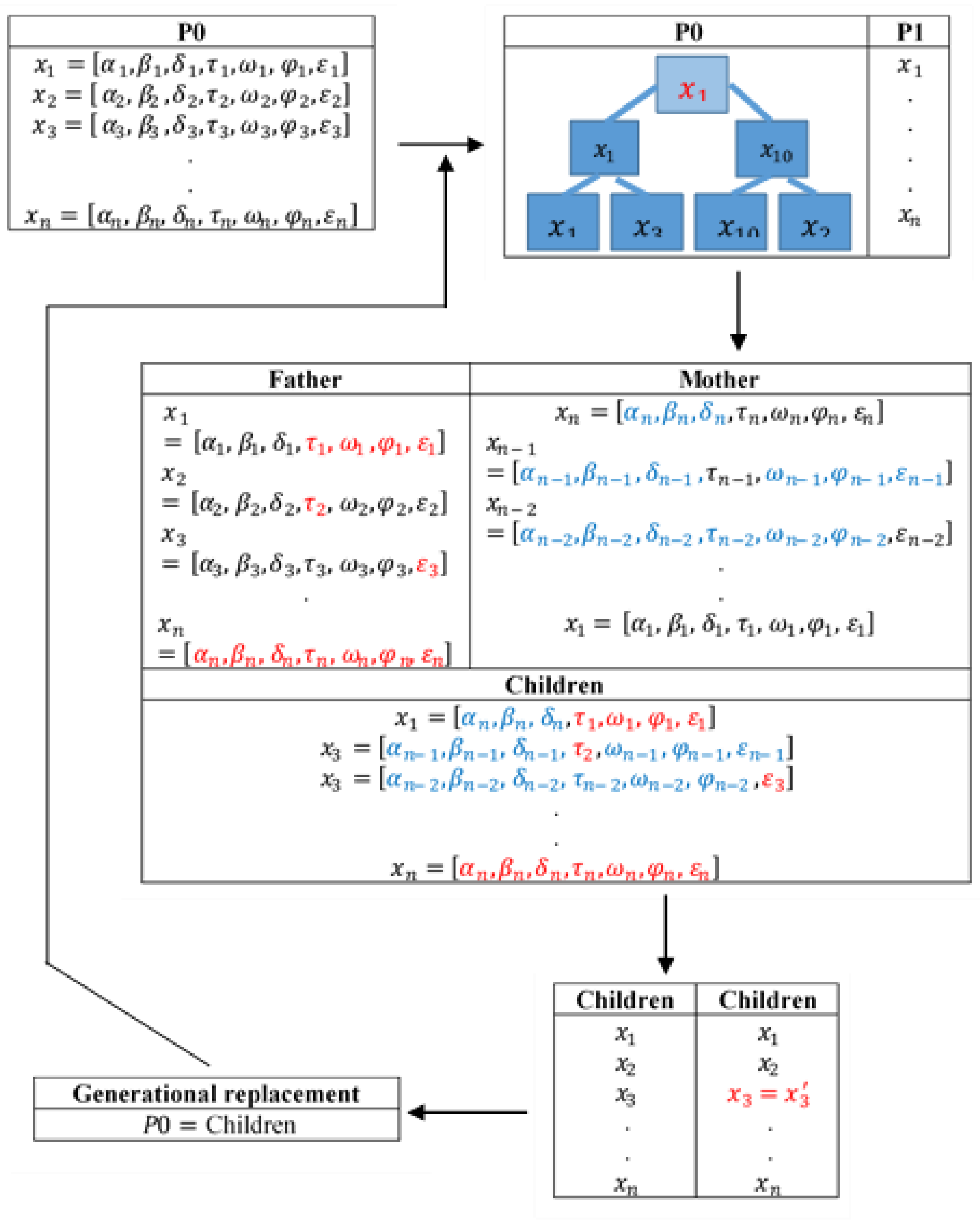

P0
$x_1 = [\alpha_1, \beta_1, \delta_1, \tau_1, \omega_1, \varphi_1, \varepsilon_1]$
$x_2 = [\alpha_2, \beta_2, \delta_2, \tau_2, \omega_2, \varphi_2, \varepsilon_2]$
$x_3 = [\alpha_3, \beta_3, \delta_3, \tau_3, \omega_3, \varphi_3, \varepsilon_3]$
$x_n = [\alpha_n, \beta_n, \delta_n, \tau_n, \omega_n, \varphi_n, \varepsilon_n]$
P0
P1
Father
Mother
Children
Children
Children
$x_3 = x_3'$
Generational replacement
P0 = Children

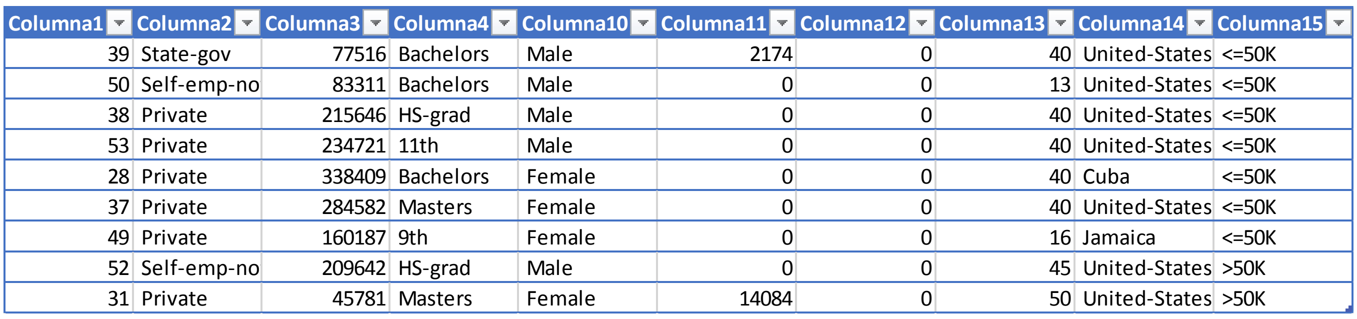
Fig. 2. Structure of the functioning of the genetic algorithm and DNN.

3. EXPERIMENT

The proposed DNN re-configuration model was implemented using the KERAS and TENSORFLOW libraries of Python 3.5.8. This model was run on a computer with an Intel® inside CORE^{TM} i3 processor and 8 GB of RAM using the eclipse interpreter. In addition, the validation of the model was made through one (1) test available in a repository, whose objective is to determine if a person earns more or less than 50,000 US per year from fourteen (14) different attributes of each subject [15].

The characteristics are part of a database that contains the records of a census, with attributes such as: age, daily work hours, the age of the companion, the position, the type of work, gender, nationality, the level of education, marital status, hours of work per week, race, capital earned and lost in the year (see Fig. 3). There are two databases one (1) with 48842 (Training = 32561, Test = 16281) instances to train the classifier (DBA) and another with 45522 (Training = 30162, Test = 15060) to validate it (DBB).

DBA has a set of continuous and error-free data in order to allow the user to propose a model or train their classifier without any problem. However, DBB is not filtered, that is, this database has instances with errors or missing information to check if the proposed model still works.



Columna1	Columna2	Columna3	Columna4	Columna10	Columna11	Columna12	Columna13	Columna14	Columna15
39	State-gov	77516	Bachelors	Male	2174	0	40	United-States	<=50K
50	Self-emp-no	83311	Bachelors	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	Female	0	0	16	Jamaica	<=50K
52	Self-emp-no	209642	HS-grad	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	Female	14084	0	50	United-States	>50K

Fig. 3. Segment of the database used.

In this work the DNN was trained with each configuration of individuals generated by the genetic algorithm. The genetic algorithm evaluates a set of 50 individuals per generation, during one hundred (100) generations and was executed one hundred (100) times in a computer. In each execution of the algorithm the individual with the best aptitude value found is stored and compared with the validation data, to verify if optimizing DNN operating resources are appropriate topologies and configurations.

4. RESULTS

During the operation of the proposed model, the aptitude value of the individuals was measured and represented by two graphs. In the first one (Fig. 4), each individual component of the aptitude value is presented during one (1) execution of the model and its behavior when unifying by the combination of linear factors during the evolution of the genetic algorithm. The second one shows (Fig. 5) the behavior of the best aptitude value found by generation during all the executions of the proposed model.

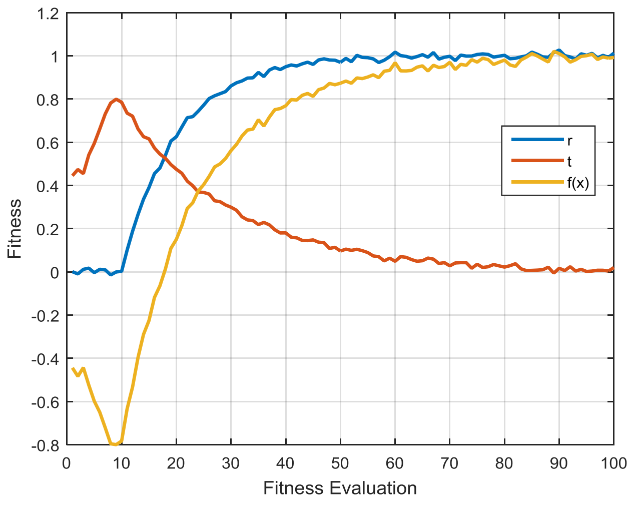


Fig. 4. Behavior and components of the best aptitude value of the population found during one (1) execution of the genetic algorithm.

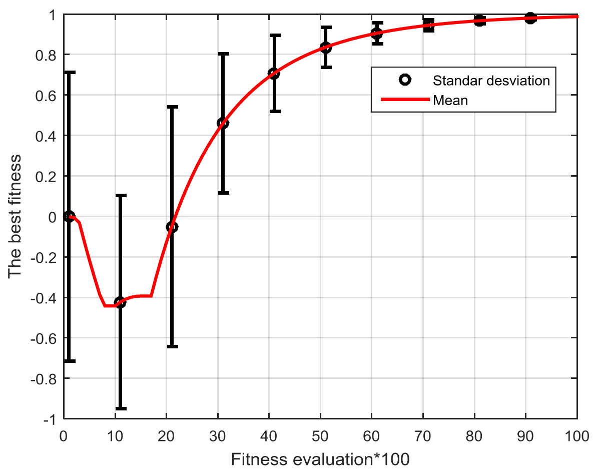


Fig. 5. Behavior of the margin of error and the standard deviation for one hundred (100) executions of the genetic algorithm.

By taking the characteristics of the best neural network model found using the optimization method, the validation was performed and estimating the average error of the DNN output. This value can be compared with others reported in the state of the art, as shown in Table 2.

Table 2. Average error obtained using other techniques compared to the proposed model [15].

Algorithm	Error
C4.5	15.54
C4.5-auto	14.46
C4.5 rules	14.94
Voted ID3 (0.6)	15.64
Voted ID3 (0.8)	16.47
T2	16.84
1R	19.54
NBTree	14.1
CN2	16
HOODG	14.82
FSS Naive Bayes	14.05
IDTM (Decision Table)	14.46
Naive-Bayes	16.12
Nearest-neighbor (1)	21.42
Nearest-neighbor (3)	20.35
OC1	15.04
AG_DNN (Proposed model)	15.03

5. CONCLUSION

In Fig. 4. It is observed that it is possible for the proposed strategy to minimize the execution time of the DNN, so that it makes the classification of a group of patterns in an adaptive way. That is, it is not necessary to know the type of data, since the proposed strategy is responsible for finding the best configuration to generalize its behavior.

One of the advantages of the proposed strategy is that it allows the user to reduce the margin of error when configuring a DNN, which allows this model to be compared with others present in the state of the art as shown in Table 2. It can be said that one possibility that the margin of error is not close to zero is that enough generations were not defined for the genetic algorithm to converge to a definitive solution. However, the advantage of this strategy is that, compared with these methods, it is possible to minimize the computational cost when executing a classifier in a computer and still perform the same task of classification as one that has a large consumption of resources.

As shown in Fig. 5. The algorithm has decreased the margin of error as it evolves. This feature gives an advantage, since having a low margin of error with good individuals it is possible to generalize it to perform signal processing in real time. This would allow a user to identify partial signal characteristics in portable devices and in different operating conditions.

REFERENCES

[1] A. E. Karnga, "Neural networks," IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339), Washington, DC, USA, 1999, pp. 4419-4421 vol.6. doi: 10.1109/IJCNN.1999.830881.

[2] C. Guojin, Z. Miaofen, Y. Honghao and Li Yan, "Application of Neural Networks in Image Definition Recognition," 2007 IEEE International Conference on Signal Processing and Communications, Dubai, 2007, pp. 1207-1210. doi: 10.1109/ICSPC.2007.4728542.

[3] C. Guojin, L. Yongning, Z. Miaofen and W. Wanqiang, "The image auto-focusing method based on artificial neural networks," *2010 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, Taranto, 2010, pp. 138-141. doi: 10.1109/CIMSA.2010.5611751.

[4] M. Juayek and R. Sotelo, "An Artificial Neural Network approach for No-Reference High Definition Video quality assessment," 2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Nara, 2016, pp. 1-3. doi: 10.1109/BMSB.2016.7521900.

[5] Y. Zhiqi, "Gesture learning and recognition based on the Chebyshev polynomial neural network," 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, Chongqing, 2016, pp. 931-934. doi: 10.1109/ITNEC.2016.7560498.

[6] W. Wang, J. Liu, F. Yang and C. Cai, "A Method of Dynamic Spectrum Access in HD Radio Based on BP Neural Network," 2014 International Conference on Wireless Communication and Sensor Network, Wuhan, 2014, pp. 180-183. doi: 10.1109/WCSN.2014.43.

[7] X. Pang, H. Ma, P. Su and G. Tang, "TPPMA: New Adaptive BP Neural Network Based on PSO and PCA Algorithms," 2018 IEEE 27th International Symposium on Industrial Electronics (ISIE), Cairns, QLD, 2018, pp. 637-642. doi: 10.1109/ISIE.2018.8433786.

[8] G. P. Zhang, "Neural networks for classification: a survey", IEEE Transactions on Systems Man & Cybernetics Part C Applications & Reviews, vol. 30, no. 4, pp. 451-462, 2000.

[9] S. C. Tan, J. Watada, Z. Ibrahim and M. Khalid, "Evolutionary Fuzzy ARTMAP Neural Networks for Classification of Semiconductor Defects," in IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 5, pp. 933-950, May 2015. doi: 10.1109/TNNLS.2014.2329097.

[10] M. A. Shafiq, "Direct adaptive inverse control of nonlinear plants using neural networks," *2016 Future Technologies Conference (FTC)*, San Francisco, CA, 2016, pp. 827-830. doi: 10.1109/FTC.2016.7821699.

[11] D. Lee, J. Lee, "Equilibrium-based support vector machine for semisupervised classification", IEEE Trans. Neural Netw., vol. 18, no. 2, pp. 578-583, Mar. 2007.

[12] Henry Hernández, Rodrigo Moreno, Andres Faina, and Jonatan Gomez, Design of a bio-inspired controller to operate a modular robot autonomously, Advances in Artificial Intelligence - IBERAMIA 2018 (Cham) (Guillermo R. Simari, Eduardo Fermé, Flabio Gutierrez Segura, and José Antonio Rodríguez Melquiades, eds.), Springer International Publishing, 2018, pp. 314–325.

[13] D. Zhang, X. Han and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," in CSEE Journal of Power and Energy Systems, vol. 4, no. 3, pp. 362-370, September 2018. doi: 10.17775/CSEEJPES.2018.00520.

[14] Terrence J. Sejnowski, "The Deep Learning Revolution," in The Deep Learning Revolution, MITP, 2018, pp.1-10.

[15] Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.