International Journal of Intelligent Computing and Information Science

Architecture Optimization Model for the Deep Neural Network

Kingsley Chiwuike Ukaoha
Department of Computer Science University of Benin
Benin City, Nigeria.
kingsley.ukaoha@uniben.edu

Efosa Igodan
Department of Computer Science
University of Benin
Benin City, Nigeria
charles.igodan@uniben.edu

Abstract

The daunting and challenging tasks of specifying the optimal network architecture and its parameters are still a major area of research in the field of Machine Learning (ML) till date. These tasks though determine the success of building and training an effective and accurate model, are yet to be considered on a deep network having three hidden layers with varying optimized parameters to the best of our knowledge. This is due to expert's opinion that it is practically difficult to determine a good Multilayer Perceptron (MLP) topology with more than two or three hidden layers without considering the number of samples and complexity of the classification to be learnt. In this study, a novel approach that combines an evolutionary genetic algorithm and an optimization algorithm and a supervised deep neural network (Deep-NN) using alternative activation functions with the view of modeling the prediction for the admission of prospective university students. The genetic algorithm is used to select optimal network parameters for the Deep-NN. Thus, this study presents a novel methodology that is effective, automatic and less human-dependent in finding optimal solution to diverse binary classification benchmarks. The model is trained, validated and tested using various performance metrics to measure the generalization ability and its performance.

1 Introduction

The adaptive mechanism that gives computer the ability to learn by examples, by analogy, and from experience is referred to as Machine Learning (ML) (Negnevitsky, 2005). The most popular approaches to ML are Artificial Neural Network (ANN) and Evolutionary Algorithm (EA). ANNs are referred to as universal approximators (Svozil *et al.,* (1997) and Qin and Tang, (2009)) with similar principle but different types and structures used to solve variety of complex problems still with limitations. The most commonly and heavily used ANN is the MLP (En *et al.,* (2008), Karsoliya, (2012), Kokko, (2013), Abdalla *et al.,* (2014), Sewsynker-Sukai *et al.,* (2017), and Geron, (2019). One major evolutionary algorithm used is the population based genetic algorithm (GA) (Goldberg, 1989). Some of the non-trivial limitations are the optimization of architectural design and model parameters. Although, these parameters are decided by trial-and-error: however there is still the need to reduce complexity and human dependency in determining best designed model as an ongoing research (Abdalla *et al.,* 2014). The daunting and challenging tasks of specifying the optimal network architecture and its parameters are still a major area of research in the field of Machine Learning (ML) till date (Igodan, 2019 unpublished). In literature, the time consuming error-and-trial rule-of-thumb method is used for determining the number of hidden units (Ferentinos, (2005) and Panchal *et al.,* (2011)). However, in some literature, optimization process still partially automated, results in several problems affecting the generalization ability (Ferentinos, (2005), Guler, (2005), Panchal *et al.,*(2011), Batchis, (2013), Abdalla *et al.,*(2014), and Sewsynker-Sukai*et al.,*(2017)), and the obvious scarcity of automated alternative procedures for optimal selection of topologies which is assumed by trial-by-error or heuristic procedures (Fisher and Leung, (1998), and Sewsynker-Sukai*et al.,*(2017)) are commonplace in most literature to the best of our knowledge. This is because of the problems of deception and multimodality in the search space of the network architectures which are usually pre-defined (Ferentinos, 2005) by trial and error approach. An MLP, a model, with no known guidelines for its structures, can approximate any arbitrary function (Karsoliya, (2012), Abdalla *et al.,* 2014, Kokko, (2013), and Sewsynker-Sukai *et al.,* (2017)). MLP's with two or more hidden layers are more efficient in approximating any input/output map (Sivanandam *et al.,* (2014) and are regarded as deep neural network. However, it is practically difficult to determine a good network topology just from the number of inputs and outputs as it depends critically on the number of training samples and complexity of the classification to be learnt (Karsoliya, 2012). The number of hidden layers is the trade-off between smoothness and accuracy. As a small number of hidden layers increase the smoothness of approximation, a greater number of hidden layers increase the accuracy of approximation (Kokko, 2013). This is because there are problems with one input and one output that require millions of hidden units, and problems with a million inputs and a million outputs that require only one hidden unit, or none at all (Karsoliya, 2012). One major evolutionary algorithms used to address these problems is the population-based Genetic Algorithm (GA). According to Goldberg (1989) genetic algorithms are population-based and search algorithms based on the mechanics of natural selection and natural genetics. GAs combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search.

In this paper, an evolutionary genetic algorithm based deep neural network (E-Deep-NN) is proposed for the prediction of prospective student's admission into the University of Benin. The model evolves three (3) hidden layers and the number of neurons employed to achieve high accuracy. By taking suitable number of hidden layers

and the number of neurons in each hidden layer, better results are achieved in less training time; and in increasing the number of hidden layers up to three (3) layers, accuracy can be achieved up to great extent if accuracy is the main criteria for designing a neural network. The evolution of the hidden layers in this study will help in the trade-off between accuracy and time complexity as the layers and neurons are evolved using the GA. The evolved number of neurons of the architecture is achieved from the proposed heuristic using $(2*n+1)$ as suggested in Kokko, (2013) and Sivanandam *et al,* (2014) and based on the number of features and instances of the datasets adopted, where n is number of input values. This study intends to combine the advantages of DNN and GA to address some of these problems, as they are considered the most reliable and promising computational intelligence techniques (Chiroma *et al.,* 2017).

2. Literature Review

In the resent years, many Genetic Algorithm (GA) based evolution model has been proposed and presented for solving varieties of problems ranging from regression problems, binary classification problems to multiclass classification problems in building an evolving one hidden layer or two hidden layer networks with varying numbers of hidden neurons (units) per layer. However, only a few literatures have implemented their works on a three hidden layers (deep networks) with similar regularization parameters. According to Geron (2019) for many problems using a Multilayer Perceptron (MLP) with single hidden layer can provide a reasonable result even for the most complex functions provided it has enough neurons. This convinced researchers that there is no need to investigate any deep neural networks. In Brownlee, (2017) deep learning refers to having many hidden layers in the neural network. They are deep because they would have been unimaginably slow to train historically, but may take seconds or minutes to train using modern techniques and hardware. However, the fact that deep networks have a much higher parameters efficiency than shallow ones are overloaded: they can model complex functions using exponentially fewer neurons than shallow nets, allowing them to reach much better performance with the same amount of training data (Geron, 2019). The large number of applications of the genetic algorithm for the artificial neural networks is exposed in any of these folds. The GA can either be used as a means to either learn Artificial Neural Networks (ANNs) connection weights that are coded in a genetic string as binary or real numbers or evolve and select the architecture and parameters together or independently from the evolution of weights. Another means is to combine both methods using GA. The encoding methods could either be the strong specification scheme (or direct encoding scheme) where a network's architecture is explicitly encoded or a weak specification scheme (or indirect scheme) where the exact connectivity pattern is not explicitly represented instead it is computed in the basis of the information encoded in the string by a suitable developmental rule (Arifovic and Gencay, 2001) previous studies related to this topic have been performed.

In Abdalla *et al.,* (2014) a new methodology for optimizing ANN parameters is developed. The objective is to design an ANN using GA direct encoding specification scheme that is less human dependence with high performance. The study achieves an acceptable degree of success with a total of 26 bits chromosome. Thomas *et al.,* (2015) developed a novel system called Heurix to predict the optimal number of hidden nodes configured to favour speed (low complexity) accuracy, or a balance between the two. A single hidden layer was adopted in the work. The system

developed produced high performance compared to the exhaustive search (ES). Nienhold, (2005) presented a modified genetic algorithm employed to optimize the Feedforward Neural Network (FNN) on an EEG dataset. A single hidden layer with randomly selected number between 3 and 25 using the Nguyen-Widrow learning algorithm is adopted in the study. The study showed optimal performance was achieved. In Ferentinos, (2005) presented two neural networks with one and two hidden layers respectively encoded using an indirect specification. The numbers of units in the hidden layers were obtained using GA. The study shows the influence of the variation of the GA parameters is not significant compared to the network parameters. Senhaji *et al.,* (2017) proposes a new multi-objective training model with constraints to satisfy two objectives: the learning objectives and number of weights and neurons optimization. The model provides balance between the MLP and the complexity to get good generalization using an evolutionary approach known as the Non-Dominated Sorting Genetic Algorithm (NSGA II). Pater, (2016) presented an optimized model for crude distillation process. In the study, the optimization of the weights and network structure showed great potential in the crude oil refinery and in general model prediction for global optimization problem. Taskiran *et al.,* (2015) presented an optimized BP-Learning algorithm and an MLP structure using three different heuristics optimization algorithms: ABC, GA and SA. Sagar and Chalam, (2011) presented an optimized fixed FNN architecture using evolutionary algorithm to find a near optimal set of connection weights. The proposed EAANN approach gave a zero mean squared error than the gradient descent method. Plagianakos *et al.* (2005) proposes an evolutionary strategy based MLPs with pure threshold activation functions. The model provides an interesting alternative class of neural networks, as it requires significantly less amount of memory for storage of weights and uncomplicated digital arithmetic operations when compared to networks with real weights and non-linear (sigmoid) activation functions. The results showed great promises. Rao and Gupta, (2014) developed a new method to design pattern detection using the combination of genetic algorithm and MLP networks. The study shows great significance to the re-engineering process designer. Hassan and Jasim (2010) applied GA on ANN for pattern recognition. The results obtained revealed a great deal of improvement than the trial-and-error approach with 96% accuracy. Batchis, (2013) presented a genetic algorithm based evolutionary ANN with a fixed architecture and connection weights. The study showed that the EANN approach leads to a promising prospect than the traditional ANN method. Castillo *et al.,* (2000b) presented the application of a GA-Prop based BP-MLP model to solve function approximation problems. In the study, the Evolutionary Algorithm (EA) selects the initial weights and learning rate, and changes the number of neurons in the hidden layers. In Idrissi *et al.,* (2016) a multi-objective mathematical formulation comprising of a GA and BP algorithm is presented in determining the optimal number of hidden layers, its neurons and weight values. The study demonstrates the effectiveness of the proposed model. Zhou and Li, (2004) adopted a GA approach to decide the optimal MLP structure. In the study, the number of nodes in each hidden layers were obtained dynamically with the synaptic connecting weights. Results show that top performance can be obtained by MLP with simpler modal structure. Fakharudin *et al.,* (2013) combined GA and ANN to optimize the production of biogas process. In the study, the evolutionary neural network is implemented to eliminate the need to determine the number of hidden nodes and evolve the best structures and initial weights. The results showed low error and increased accuracy by 0.44% in the biogas yield. Balochion *et al.,* (2013) optimized the MLP classifier using GA for the classification of audio to speech and music. The classifier

model adopted the wavelet transform model for the selection of features and showed improvement accuracy of 96.49%. In Kumari and Kumar (2015) a multilayer perceptron neural network optimization using GA to classify ECG arrhythmia is presented. In the study, the learning rate and momentum term were optimized to improve the classifier model. Results show high classification accuracy; average precision and average recall of 96.93% an d96.92% respectively. Ganatra *et al.,* (2011) proposed a hybrid system for weight optimization in BP-ANN using GA to improve performance of the classifier. The study shows the merits of combining BP and ANN. Ettaouil *et al.,* (2015) presented an optimized architecture MLP based model to obtain an optimized number of hidden layers and units depending on the dataset. Ludermir *et al.,* (2006) presents a methodology for the neural network global optimization to simultaneously optimize the MLP weights and architectures to generate few connections with high classification performance for any datasets. The approach combines the SA, TS and BP training algorithms and showed improvement in accuracy. Alejandro and Andres (2011) proposes an evolutionary algorithm based MLP model using GA and Binary PSO to improve the predictive power of a credit risk score card. Results obtained shows that both methods outperform the logistic regression and default neural networks in terms of predictability. However, the BPSO prove to be less time consuming than the BPS. In Aghazadeh and Gharehchopogh, (2017) the optimization of the MLP using genetic algorithm is presented. The study provides a hybrid approach to estimate the cost of websites design by content management system (CMS). Guler *et al.,* (2015) proposes an optimal MLP using GA to search for optimal structure and training parameters for better prediction for lung sound. The study resulted in the design of optimal network structure and improved reduction in the processing load and time. Jayaraj and Ramin, (2016) propose the hybridization of MLP with GA to optimize the parameters of the model to build an optimal model selection method for software defect prediction (SDP). The model shows improvement in classification accuracy by 0.61%. Castillo *et al.,* (2000a) propose a new hybrid approach of combining a single hidden layer MLP with an evolutionary algorithm and BP algorithm. The experiment shows that the proposed model achieved better results and obtained optimal MLP network with high classification accuracy than conventional procedures. Rahman and Setu, (2005) implemented a collaborative approach for combining GA and ANN into a single system to find optimal model using MATLAB 7.0.12. The study shows potential and good performance. Chow *et al.,* (2001) implemented a one-hidden layer and two-hidden layer FNN using LM algorithm for chiller system. The study demonstrates the ability of the model to learn complex non-linear mapping and find global optimal solution to problems. The result shows that the 2-hidden layer with 5-5-9-4 configurations produces better performance with a MSE<0.01 than the one-hidden layer configuration. Carvolho *et al.,* (2010) proposed an evolutionary ANN in a systematical and automatic way by adopting four meta-heuristics: generalized external optimization, variable neighbourhood search, SA and canonical GA for model selection. The approach evolves the number of hidden layers, units in each hidden layer, the learning rate, momentum term and activation functions. The results show better performance than human specialists.

3. Multilayer Perceptron, Deep Neural Network and Backpropagation

An MLP is composed of one input layer, one or more layers of Threshold Logic Units (TLUs), called hidden layers, and one final layer of TLUs called the output layer. The layers close to the input layer are usually called the lower

layers, and the ones close to the outputs are usually called the upper layers. Every layer except the output layer includes a bias neuron and is fully connected to the next layer. When an ANN contains a deep stack of hidden layers, it is called a Deep Neural Network (DNN). The field of Deep Learning studies DNNs, and more generally models containing deep stacks of computations. Even though Deep Networks have much higher parameter efficiency than shallow ones, they can model complex functions using exponentially fewer neurons than shallow nets, allowing them to reach much better performance with the same amount of training data (Geron, 2019). To train an MLP (or DNN) the backpropagation training algorithm (or simply Gradient Descent) is used for automatic computation (i.e. forward and backward passes through the network). In other words, it is a process of reducing the error function by tweaking the connection weight and biases. This process is repeated until the network converges to the optimal solution. The performance of neural network depends not only on problem and purpose of the network also on the complexity of the input datasets. There are many performance measures that can be used like the error functions of MSE, RMSE, SSE, and so on; the Receiver Operating Curve (ROC) and Confusion Matrix and their usage is not unambiguous. In modeling an MLP or DNN, three stage operator are commonly adopted which are training, validation and testing. The model is fit during training using training dataset, validated using the validated dataset and evaluated using the testing datasets on how well the model generalizes on the unknown datasets. The low the generalization error is the fitter the model. Figure 1 depicts the network architecture for a deep NN.

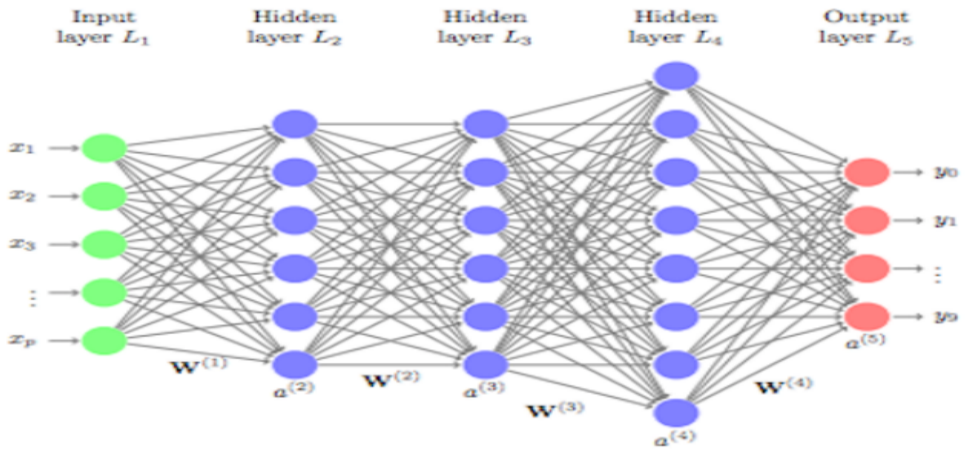


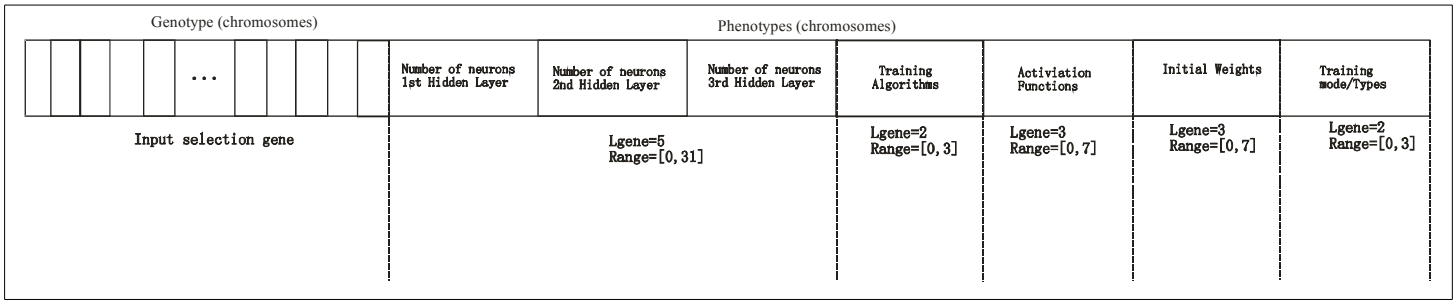
Figure 1: Structural representation of a Deep Neural Network with three hidden Layers

4. Genetic Algorithm

The combination of optimization techniques and general purpose optimization methods based on Darwin Theory of Evolution that searches for optimal solutions of a complex objective function by simulation of the evolutionary process is called Genetic Algorithm (GA) and it is applied in a variety of problem domains (Goldberg, 1989). The GA makes use of variation operators: selection, crossover and mutation operators. The GA algorithm begins with multiple solutions of a given problem being examined. According to Ferentinos, (2005) the main aspect in the evolutionary process is the way of representing the solution (phenotype) by encoding it into a specific genotype and

grouped together to form chromosomes which makes up the population of the problem. A genotype is a sequence of bits (0 or 1) with a specific constant length and each corresponds to a unique phenotype and this representation treats the problem as a combinatorial one. A phenotype in this study consists of the Deep Neural Network topology, its activation functions, the training algorithms, the initial weights and the training modes. The training modes indicate the batch (offline), mini-batch and sequential (online) training modes. The chromosome structure is represented in Figure 2.

Figure 2: Chromosome Structure



5. Methodology

This paper aim to address the problems in designing of a DNN when using the trial-and-error procedure by adopting a methodology based on evolutionary GA method. The optimization involves the network parameters which include the architecture, activation function, training algorithm, initial weights and training modes. The detail steps of the proposed evolutionary genetic based artificial neural network is depicted in Figure 3.

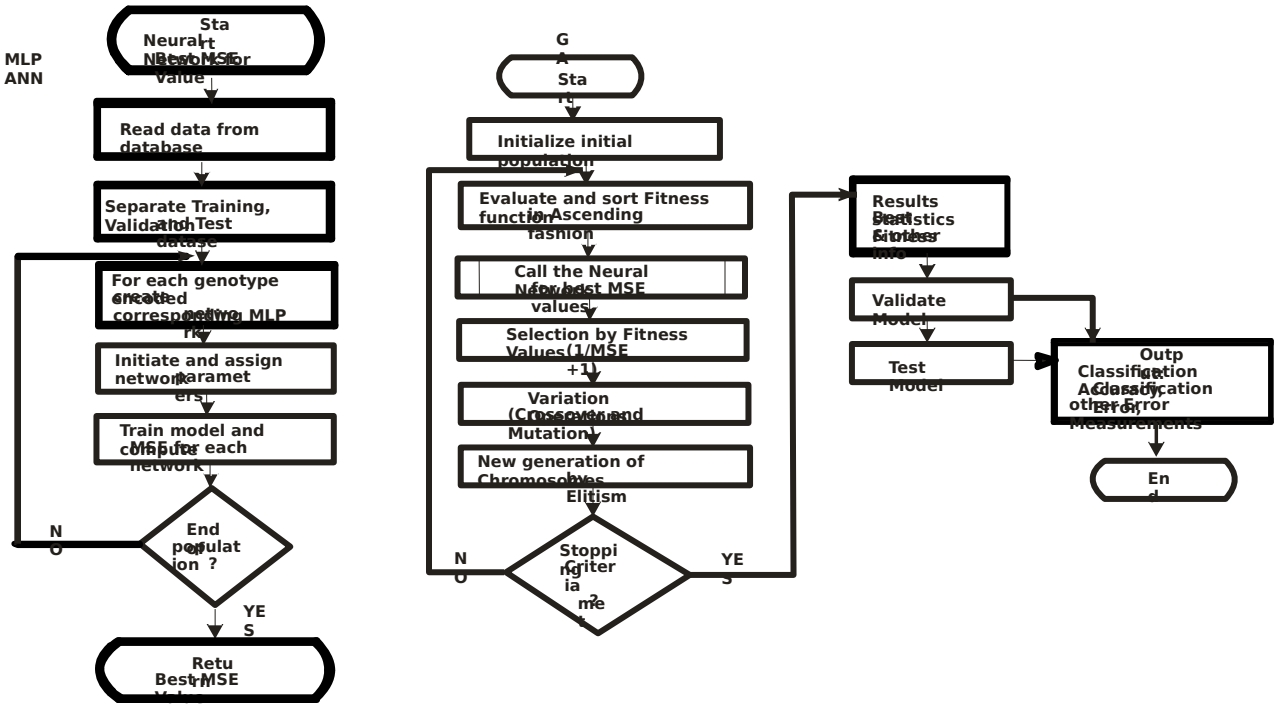


Figure 3: Proposed Model Process

5.1. Data Collection

The benchmark dataset used in this paper was collected from the admission office personnel of the University of Benin. The historical dataset ranges from 2015 to 2017 sessional admission periods. However, 2000 records comprising 61.75% (1235) admitted cases and 38.25% (765) non-admitted cases were selected and normalized in this study. Also, the standard XOR dataset is used in this study for testing the model.

5.2. Preparations and scaling of the dataset

In this study, the following preprocessing steps is adopted purposely to eliminate outliers, inconsistencies, and incompleteness of data in order to make the data suitable for mining so as to achieve better accuracy (Tan, *et al.,* (2006) and Gorunescus, (2011)): Data Cleaning: this step involves physically preparing the data by intentionally removing irrelevant data to reduce as much noise as possible so as to guarantee accuracy and validity of the data. Data Integration: suspected redundant (repeated) and inconsistent data are removed from the dataset. Normalization/Standardization: according to Ashwood, (2013) and Brownlee, (2017) machine learning perform better when the dataset fed into it are standardized between 0 and 1. The data are scaled so that the input series would all have same magnitudes and due to the way activation functions approximate (squash) the input data (Ashwood, 2013). Neural networks require the input to be scaled in a consistent way to the range between 0 and 1 called normalization (Brownlee, 2017). In this study, Microsoft Excel is used in the normalization of the dataset by adopting the MIN-MAX model as represented in Equation 1.1. Data mining: After applying the previous steps, data mining techniques (genetic optimization and neural network classification) were applied on the datasets to extract desirable knowledge.

$$X' = (max_2 - min_2)\frac{x - min_1}{max_1 - min_1} + min_2 \quad \ldots 1.1$$

where X' refers to the normalized value, x refers to the observation value (original value), min_1 and max_1 are the respective minimum and maximum values of all observations, and min_2 and max_2 refer to the desired minimum and maximum of the new scaled series.

5.3. DNN Development Model

A deep neural network with three (3) hidden layers, an input and an output was built. The input has 6 input units which represents the six features of the data sets. Since the problem is a binary classification problem, the output has one output unit representing whether the output is admitted case or non-admitted case. The three hidden layers comprise of varying number of units based on the resultant GA evolution and algorithm represented in Figure 4. The second evolved model is used to predict the outcome of the XOR problem with two input units and one output unit.

5.4. DNN-GA Combination

The combination of DNN and GA technique shows the modeling and optimization processes of a DNN that can model complex functions using exponentially fewer neurons with better performance.

5.4.1. GA encoding Scheme

In this work, the DNN design and training parameters were represented using the indirect encoding specification scheme in encoding the chromosome into specific genotype. A genotype represents a sequence of bits with specific constant length corresponding to a chromosome.

5.4.2 Bit-string representation

In this study an indirect (Binary encoding scheme) representations is adopted. The scheme incorporates five tasks of the DNN design and training parameters:

(i) The selection of the layers and its units
(ii) The selection of the training algorithms
(iii) The selection of the activation function types
(iv) The initialization of the initial weights
(v) The selection of the training modes

5.4.2.1 The network architecture encoding

The various sizes of the hidden neurons are evolved using GA as depicted in Figure 4. That is, using the mathematical formulation suggested by Sivanandam *et al* (2014), and Kokko, (2013) given as (2n+1), where n is the number of features used to determine the number of neurons in the first hidden layer first before calculating for the second and third layers where the arises. The total number of hidden neurons is formulated based on the size of the instances and number of features, m ($2^{\text{number of features-1}}$ - 1) from the datasets. The algorithm for implementing the determination of the number of neurons in the hidden layers is shown in Figure 1.3. The binary entry of bit strings is represented from 0-4 for the three hidden layers having a gene size of 5bits.

5.4.2.2 The minimization (training) algorithm encoding

The classical and still preferred training algorithms for neural networks are still the stochastic gradient descent (Brownlee, 2017). In this study, four stochastic training algorithms are adopted for training the proposed model: Levenberg-Marquardt, conjugate gradient, quasi-Newton, and steepest descent algorithms. They are represented with a gene size of two in the fifth and sixth bits of the binary string (chromosome) representation.

5.4.2.3 Activation (Transfer) Function encoding

The choice of activation function is strongly constrained by the type of problem that is being modeled. However, for pattern recognition problems typically the Sigmoid functions: Log-Sigmoid or Tangent-Sigmoid is used (Hagan *et al.*, 2014). The non-linear activation function is critical for the power of neural network (Manavazhahan, 2017) and has high aptitude to simulate extreme data (Dorofki, *et al.,* 2012). In this study, two non-linear, Log-Sigmoid-based

activation functions are adopted which is represented in Equations 1.2 and 1.3 respectively. With a gene size of three, the activation function occupies the seventh to the ninth bits string in the chromosome structure.

$$f(x)=\frac{1}{1+e^{-x}} \quad \ldots 1.2$$

$$f(x)=\tanh(x)=\frac{2}{1+e^{-2x}}-1 \quad \ldots 1.3$$

```
if phenotypic architecture (npa) >= 20 and the no of instances <= 2000 then
        no of 1HL = 2n+1
        a = npa - 1HL
        if (a is Even) then
                no of 2HL = 3HL = a/2
        else
                b = truncate (a/2)
                c = b
                no of 2HL = b+1
                no of 3HL = c
        End if
else
  if phenotypic architecture (npa) < 20 then
        no of 1HL = 2n+1
        if npa<= no of 1HL then
                create one hidden layer (1HL)
        else
                1HL = 2n+1
                2HL = npa - 2n+1
        End if
End if
```

Figure 4: Algorithm for the determination of the number of neurons in the hidden layers

5.4.2.4 Weight Initialization encoding

Weights are often initialized to small random values, such as values in the range 0 to 0.3, although more complex initialization schemes can be used. Like linear regression, larger weights indicate increased complexity and fragility of the model (Brownlee, 2017). This is because large weights are disadvantageous for the neuron since they will lead to a high output variance (Urban, 2017). However in this study, due to the fact that there is no known mathematical standard or range of weighted values for the weight initialization, five different weights ranges from the works of Arifovic and Gencay (2001) and Kokko (2013) were adopted. With a gene size of three, the initial weight occupies positions ten to twelve bits string in the chromosome structure.

5.4.2.5. Training modes encoding

The three different training modes are used in this study. It involves the on-line (or incremental), offline (or batch) and min-batch modes. The last two binary strings represent the different types of training modes occupying the 13th-14th-bit order in chromosome.

The summary of the chromosome is shown in Table 1.1

Table 1.1: Genetic Algorithm encoding of a MLP parameters

Bits	Meaning
1-5	Network architecture
6-7	Training Algorithm
8-10	Activation (Transfer) Function
11-13	Weights
14-15	Training modes

6.0 The Algorithm of the Proposed Model

The algorithm of the proposed system for DNN design and training parameterization consists of four main parts: the 'Users level', the 'genetic algorithm optimization' part consisting of the encoding and decoding, and the DNN 'training' part. The first part deals with the human interfacing, while the other three parts which contain several sub-sections; interacting with others to complete the procedure. The schematic representation of the algorithm is shown in Figure 5. Each box in the Figure represents a separate function and the names of these functions are inscribed on it. The interactions between functions are shown with the appropriate arrows. Explanation of the algorithm and the symbols involved follows thus:

Initially, the user initializes the parameters of the GA algorithm, The GA parameters given as the number of generations of the GA (Ng), the size of population of the AG (Ps) and finally the probabilities of the crossover (Pc) and mutation (Pm) is shown in Table 1.2 An initial population of specific random strings (chromosomes), each of which represents a network topology and other set of training parameters is generated and represented as $(X_{initial})$. The five parts encoded are the training algorithm (algo), the network architecture (archt), the activation function (actvfn), the initial weights (initwt), and training mode (trainm). This explicit information after decoding goes to the 'train' function. At this stage, each network topology with the explicit training parameters is trained with the appropriate training algorithms. The decoded genotype for the weight determines the weight's range to be randomly selected from the optional weights (rgenWt). While the learning rate (η) is determined randomly (rgenLr) within the range [0, 1], the momentum (α) is deduced using $\eta+\alpha\approx 1$ as suggested in Kokko (2013). Also, the choice of selection, crossover, and mutation methods is randomly chosen at this stage. The mean squared error (MSE) of each individual string after training is calculated and sent to the 'fitness' function where the fitness of each string is calculated (fitnessVect). The fitness is simply the value that GA tries to maximize. Here, the fitness of each individual string is given by the formula in Equation 1.4.

$$fitnes=\frac{1}{MSE+1}\ldots 1.4$$

where MSE is mean squared error for the network and the smaller its MSE the closer a fitnessvalue to 1. The selection process of the GA (function 'Select') selects the new group of strings based on fitness value, which constitute the parents of the next generation of the GA (X_{parent}) . Each string is assigned a probability of reproduction and is selected according to this probability. The probability is usually proportional to the fitness of each string. These strings are then subjected to the evolutionary operators of crossover and mutation and Elitism,

after which the final population is formed $(\overset{X}{¿¿new}\underset{¿}{})$. This process repeats until the maximum number of generation (Ng) is reached or convergence is achieved. After that, the best string, that is the string that gave the maximum fitness (or, the minimum MSE), is returned to the user, together with its corresponding minimum (best MSE) and some other information useful for statistical analysis. Conventionally, genetic algorithms with better individuals have higher fitness, corresponding to the minimum MSE of DNN adopted in this study. The GA will

terminate if any of the stopping conditions is met. The pseudocode of the evolutionary process is given in Figure 6. The GA has stopped and the relevant statistical information obtained, the best fit chromosome is decoded and the information representing the various network architecture and parameters are used to build the DNN for training, validation and testing. During the training, the three stages of evaluating the DNN used are: training, validation and testing. The training is aimed at determining the best fit model which is validated using the validation dataset. Thereafter the testing is carried out to determine how well the model chosen generalizes. By including these techniques in the system, the GA system is aimed at designing the evolutionary GA-based Deep Neural network that can be applied on any binary classification problems towards achieving high performance model.

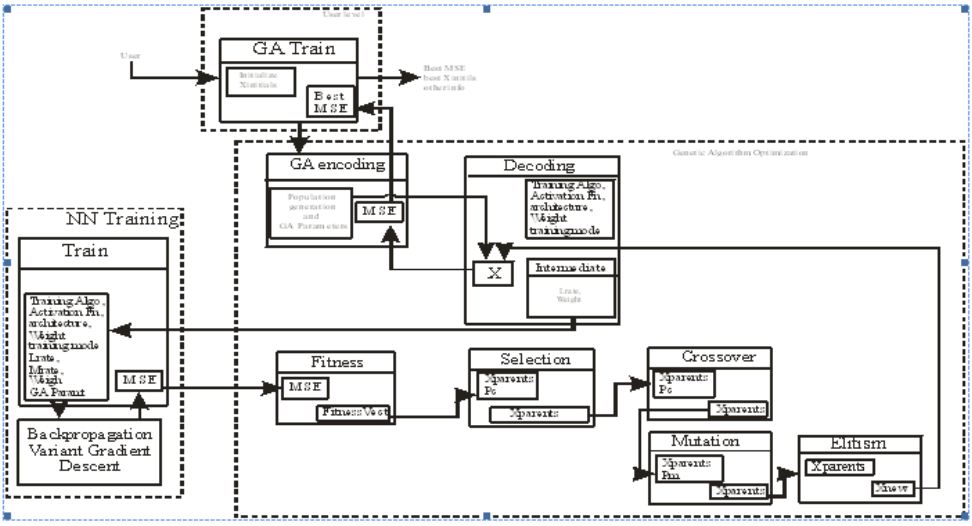


Figure 5: Schematic Representation of the Evolution Artificial Neural Network

Table 1.2: GA Parameters

Operators	Methods	Values
Population Size		[50]
Max generation		By Convergence
Selection	Roulette Wheel	Prob. 0.7
Crossover (Pc)	2-Points	[0,1]
Mutation (Pm)	Bitwise Flip	[0.6,1]
Chrom. Size		15bit string (alleles)

Initialize the population
Calculate individual Fitness function
While NOT*(termination condition) **do***
For *each genotype **do***
    Decode each chromosome of the corresponding MLP
    Perform training and calculate MSE of individual MLP
End for
Save the best individual as the best-so-far individual with the best (lower) MSE value.
    Perform Roulette-Wheel selection using Equation 3.4
    Apply genetic variation operators
    Perform Reproduction by Elitism procedure
Until (Max. time or Convergence)
    Save statistics of Results obtained

Figure 6: Pseudocode for Evolutionary Process

7.0. Results and Discussion

In this paper, a new methodology that involves combining genetic algorithm and Deep Neural Network (GA-Deep-NN) in determining the optimal network parameters has been developed. The GA-Deep-NN model has been tested for the classification of students' admission dataset of the University of Benin, Benin City, and the standard XOR dataset against those obtained by ordinary ANNs and Deep-NN using the trial-and-error procedures. The training, validation and testing dataset for the Deep-NN classification model part were feed into the evolutionary Genetic Algorithm (EA) part. The GA parameters that were adopted to classify the results are given in Table 1.2.

7.1 GA-Deep-NN Model Predictive Results for the admission dataset

In using the proposed model for the prediction of admission dataset, the best solution found is given in the following string:

101100000101100

The string is interpreted as: a deep neural network of two hidden layers network with 13 neurons in the first layer with sigmoid activation function and 9 neurons in the second layer with tangent activation function and sigmoid activation function in the output neuron, batched trained with Levenberg-Marquardt back-propagation training algorithm that uses [-0.5,0.5] as the initial weight range. The solution gave a value of 0.00032 after 500 epochs

The second best solution found gave the following string:

110010000100110

The string is interpreted as: a deep neural network of three hidden layers network with 13 neurons in the first layer with sigmoid activation function and 9 neurons in the second layer with sigmoid activation function, 3 neurons in the third layer with hyperbolic tangent activation function and sigmoid activation function in the output neuron, sequentially trained with Levenberg-Marquardt back-propagation training algorithm that uses [-0.125, 0.125] as the initial weight range. The solution gave a value of 0.00049 after 800 epochs.

7.2. GA-Deep-NN Model Classification Result for the XOR dataset

The best solution found is represented in the following string:

010011011110000

The string is interpreted as: a single hidden layer neural network with 9 neurons in the hidden layer with hyperbolic tangent activation function and sigmoid activation function in the output neuron, batched trained conjugate gradient back-propagation training algorithm that uses [-1. 1] as the initial weight range. The solution gave a value of 0.00018 after 300 epochs.

The second best solution found is represented in the string as:

011000100001000

The string is interpreted as: a single hidden layer neural network with 12 neurons in the hidden layer with sigmoid activation function and sigmoid activation function in the output neuron, batched trained steepest descent back-propagation training algorithm that uses [-0.3, 0.3] as the initial weight range. The solution gave a value of 0.00028 after 500 epochs.

8.0. Comparative Study

The classification performance of the proposed GA-Deep-NN approach is compared to the existing ANN and Deep-NN approaches and the statistical details are given in Table 1.3.

Table 1.3: Performance statistical analysis

Metrics	GA-Deep-NN	Deep-NN	ANN
Accuracy	98.28%	97.05%	89.09%

A comparison of the results achieved by the proposed GA-Deep-NN approach against the ordinary ANN and Deep-NN shows clearly that the proposed model shows clearly that the proposed model achieved a significant degree of successes. Some of the ROC curves for the single hidden layer models showing the convergence rates for the training and testing datasets are represented in Figures 6 and 7, while the error rate and the performance of the model in terms of fitness function are depicted in Figures 8 and 9 respectively.

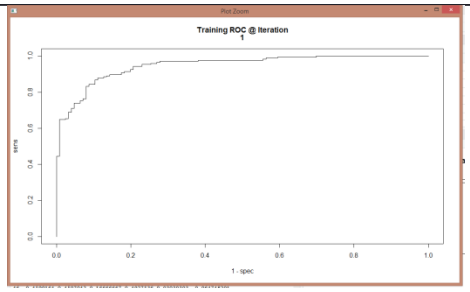


Figure 7: Convergence for Training dataset

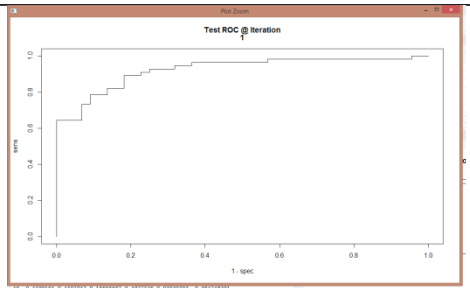


Figure 7: Convergence for Test dataset

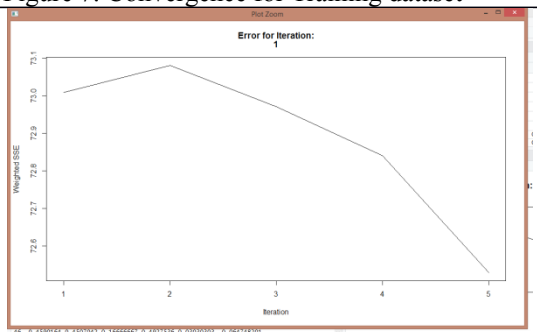


Figure 8: Error rate

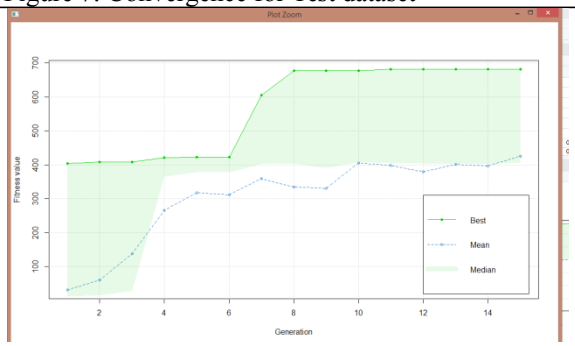


Figure 9 Model fitness performance

9.0 Conclusion and Future Work

The determination of optimal Deep-NN parameters using Evolutionary genetic algorithm has been presented in this study. The purpose of this study is to design a Deep-NN with satisfactory performance with reduced human dependency. Two datasets have been collected, treated, analyzed, and normalized for machine learning. The GA has been applied using indirect encoding representation scheme on the Deep-NN model parameters for the design and training. R programming language have been used with other machine learning libraries to obtain the results which shows that the new proposed model can optimize Deep-NN parameters precisely and effectively for better classification accuracies for binary classification problems. The aim of this study is to obtain optimal number of hidden layers, and number of hidden neurons and other training parameters like the activation functions, training algorithms, best initial weights among others, and the training modes depending on the type of dataset used – in this case, binary classification problems. This study shows that though the number of these layers is domain-specific, it is still a challenging task to obtain the perfect number of hidden neurons for all problems in advance. As a future direction of this study, the GA will be applied to determine and study the impact of the search space on time complexity, performance, and how the training time can be reduced considerately and still achieve high accuracies for both the binary and multiclass classification problems.

Reference

Abdalla O.A., Osman A., and Mohammed Y. (2014) Optimizing the multilayer Feed-Forward Artificial Neural Networks Architecture and Training Parameters Using Genetic Algorithm. International Journal of Computer Applications Vol.99, No.10, 42-48.

Aghazadeh M. and Gharehchopogh S. (2018) A New Hybrid Model of Multi-layer Perceptron Artificial Neural Network and Genetic Algorithms in Web Design Management Based on CMS Journal OF AI and Data Mining, Vo. 6 No. 2, pp. 409-415.

Alejandro, C.B., and Andres G.M. (2011) Evolutionary algorithms for selecting the architecture of an MLP neural network: A credit scoring case. In: Proceedings of the 11th IEEE International Conference on Data Mining Workshops. Vancouver, Canada, pp 725–732.

Arifovic, J. and Gencay R. (2001) Using Genetic Algorithms to select Architecture of a Feedforward Artificial Neural Network Physica A 289 pp 574-594.

Ashwood, A.J. (2013) Portfolio Selection Using Artificial Intelligence. Ph. D Thesis

Balochion, S., Seidabad, E.A., and Rad, S.Z. (2013) Neural Network Optimization by Genetic Algorithms for the Audio Classification to Speech and Music International Journal of Signal Processing, Image processing and Pattern Recognition, Vol. 6, No. 3, pp. 47-54

Batchis, P. (2013) An Evolutionary Algorithm for Neural Network Learning Using Direct Encoding Resource 53, Chinese Digital Library, Available Online: www.cs.rutgets.edu/~mlittman/courses/ml03/iCML03/.../batchis.pdf. Accessed 25th August, 2018

Brownlee, J. (2017). Deep learning with Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras. Melbourne, Australia.

Castillo, P.A., Carpio, J., Merelo, J.J., Prieto, A., and Rivas, V. (2000a) Evolving Multilayer Perceptrons Neural Processing Letters 12: 115-127

Castillo, P. A., Arenas,M. G., Castellano, J. G.,Cillero, M.,Merelo,J. J., Prieto,A., Rivas, V., and Romero,G. (2000b). Function approximation with evolved multilayer perceptrons. In

Carvalh, A.R., Ramos F.M., and Chaves A.A. (2010) Metaheuristics for the Feedforward Artificial Neural Network (ANN) Architecture Optimization Problem Neural Computation & Application

Chiroma, H., Noor, A.S.M., and Abdulkareem, S., Abubakar, A.I., Hermawan, A., Qin, H., Hamza, F., and Herawan, T. (2017). Neural Networks Optimization Through Genetic Algorithm Searches: A Review. Appl. Math. Inf. Sci. 11, No. 6, 1543-1564.

Chow T.T., Lin Z., and Song C.L. (2001) Applying Neural Network and Genetic Algorithm in Chiller System Optimization. Seventh International IBPSA conference, 13-15

Dorofki, M., Elshafie, A.H., Jaafar, O., Karim, O.A., and Mastura, S. (2012). International Conference on Environment, Energy and Biotechnology IPCBEE vol.33. pp. 39-44.

Eng M.H., Li Y., Wang Q., and Lee T.H. (2008) Forecast Forex with ANN Using Fundamental Data International Conference on Information Management, Innovation Management and Industrial Engineering. Pp. 279-282.

Ettaouil, M., Lazaar, M, and Ghanou, Y. (2005) Architecture Optimization Model for the Multilayer Perceptron and Clustering Journal of Theoretical and Applied Information Technology. Vol. 47, No. 1, pp. 64-72.

Fakharudin, A.S., Sulaiman, M.N., Salihon, J., and Zainol, N. (2013) Implementing Artificial Neural Networks and Genetic Algorithms to Solve Modeling and Optimization of Biogas Production Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013 paper No. 088, pp. 121-126

Ferentinos, K.P. (2005) Biological Engineering Applications of Feedforward Neural Networks Designed and Parameterized by Genetic Algorithms. Elsevier. Pp. 934-950.

Fischer, M.M., and Leung, Y. (1998) A Genetic-Algorithms Based Evolutionary Computational Neural Network for Modeling Spatial Interaction Data. European Regional Science Association 38th European Congress in Vienna, Austria Pp.1-25

Ganatra, A., Kosta, Y.P., Panchal, G., and Gajjar, C. (2011) Initial Classification Through Back propagation in a Neural Network Following Optimization Through GA to Evaluate the Fitness of an Algorithm. International Journal of Computer Science & Information Technology (IJCSIT), Vol. 3, No. 1, pp. 98-116.

Geron, A. (2019). Hands-On Machine Learning with Scikit-Learning, Keras, and Tensor Flow: Concepts, Tools and Techniques to Build Intelligent Systems. 2nd Edition O'Reilly

Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning Addison Wesley

Gorunescus, F. (2011) Data Mining: Concepts, Models and Techniques Springer

Guler, I., Polat, H, and Ergun, U. (2005) Combining Network and Genetic Algorithm for Prediction of Lung Sounds, Journal of Medical Systems, Vol. 29, No. 3, pp.217-231.

Hassan, A.K., and Jasim, S.S. (2010) Integrating Neural Network with Genetic Algorithms for the Clssification Plant Disease. Eng. & Tech, Journal, Vol. 28, No. 4, pp. 686-702.

Idrissi, J., Ramchoun, H., Ghanou, Y., and Ettaouil, M. (2016) Genetic algorithm for neural network architecture optimization In 2016 3rd International Conference on Logistics Operations Management (GOL), pp. 1–4.

Jayaraj, V, and Raman, S. (2016). A Genetic Algorithm Optimized Multilayer Perceptron for Software Defecr Prediction. International Journal of Advanced Technology in Engineering and Science, Vol., 4, Issue 2, pp.132-141.

Ludermir, T.B., Yamazaki, A., and Zanchettin, C. (2006) An optimization methodology for neural network weights and architectures. IEEE Transaction on Neural Networks, vol. 17, no. 6, pp. 1452-1459.

Igodan, C.E. (2019) Optimization of a Feed-Forward Neural Network Topologies and Parameters. M.Phil. Thesis. (Unpublished)

Kokko, T. (2013) Neural Networks for Computationally Expensive Problems. Master's Thesis

Karsoliya, S. (2012) Approximating Number of Hidden Layer Neurons in Multiple Hidden Layer BPNN Architecture. International Journal of Engineering Trends and Technology Volume 3, Issue 6, pp. 714-717.

Kumari, V.S.R., and Kumar, P.R. (2015) Optimization of Multilayer Perceptron Neural Network Using Genetic Algorithm for Arrhythmia Classification. Communications, Vol. 3, No. 5, pp. 150-157.

Negnevistsky, M. (2005) Artificial Intelligence: A Guide to Intelligent Systems Addison-Wesley, Harlow

Nienhold, D., Schwab K., and Hanne T. (2015) Effects of Weight Initialization in a Feedforward Neural Network for Classification Using a Modified Genetic Algorithm, 3rd International Symposium on Computational and Business Intelligence Pp 6-12.

Manavazhahan, M. (2017) A Study of Activation Functions for Neural Networks. Computer Science and Computer Engineering Undergraduate Honours Thesis

Panchal, G., Ganatra, A., Kosta, Y.P., and Panchal, D. (2011) International Journal of Computer Theory and Engineering, Vo. 3(2), pp.332-337

Pater, L. (2016). Application of Artificial Neural Networks and Genetic Algorithms for Crude Fractional Distillation Process Modeling, 2016 arXiv preprint arXiv: 1605.00097

Plagianakos, V.P., Magoulas, G.D., and Vrahatis, M.N. (2005) Evolutionary Training of Hardware Realizable Multilayer Perceptrons Neural Computing &Applic., Springer-Verlag.

Rahman, M., and Setu, T.A. (2015) An Implementation for Combining Neural Networks and Genetic Algorithms, International Journal of Computer Science and Technology Vol.6, Issue 3, 218-222.

Rao, R.S., and Gupta, M. (2014) Design Pattern Detection by Multilayer Neural Genetic Algorithm International Journal of Computer Science and Network, Vol. 3, Issue 1

Sagar, G.V.R and Chalam, S.V. (2011) Evolutionary Algorithm for Connection Weights in Artificial Neural Networks. International Journal of Electronics and Communication Engineering, Vol. 4, Number 5, pp. 517-525.

Senhaji, K., Ramchoun, H., and Ettaouil, M. (2017) Multilayer Perceptron: NSGA II for a New Multi-Objective Learning Method for Training and Model Complexity Journal of Electronic Systems, Vol. 7 No. 4, pp. 105-114.

Sewsynker-Sukai, Y., Faloye, F., and Gueguim, E.B. (2017) Artificial Neural Networks: An Efficient Tool for Modeling and Optimization of Biofuel Production (a mini review) Biotechnology and Biotechnological Equipment, 31:2, 221-235

Sivanandam, S.N, Sumathi, S., and Deepa, S.N. (2014) Introduction to Neural Networks Using MATLAB 6.0 McGraw Hill Education, India.

Svozil D., Kvasnicka V., and Pospichal J. (1997) Introduction to Multi-Layer Feed-forward Neural Networks Chemo metrics and Intelligence Laboratory Systems 39, pp. 43-62

Tan, P-N., Steinbach, M., and Kumar, V. (2006) Introduction to Data Mining Pearson

Taskiran M., Cam Z.G., and Kahraman (2015) An Efficient Method to Optimize Multi-Layer Perceptron for Classification of Human Activities International Journal of Computing, Communications & Instrumentation Eng. (IJCCIE) Vol. 2, Issue 2, pp. 191-195.

Thomas, A.J., Miltos, P., Simon, D.W., Saeed MG., and Robert, E.M. (2015) On Predicting the Optimal Number of Hidden Nodes. International Conference on Computational Science and Computational Intelligence Pp. 565-570.

Urban, S. (2017) Neural Network Architectures and Activation Functions: A Gaussian Process Approach. Thesis

Zhou, J., and Li, L. (2004) Using Genetic Algorithm Trained Perceptrons with Adaptive Structure for the Detection of Premature Ventricular Contraction, Computers in Cardiology, pp. 353-356.