Artificial Immune System for Fuzzy Backpropagation Neural Networks Optimization

Fathi Gasir
School of Applied Sciences and Engineering
The Libyan Academy for Postgraduate Studies
Tripoli, Libya
Fathi.Elgasir@Academy.edu.ly

Hiefa Alajail
Faculty of Education
Misurata University
Misurata, Libya
H.Alajail@edu.misuratau.edu.ly

***Abstract*— Fuzzy Neural Networks (FNNs) enhance conventional Artificial Neural Networks (ANNs) by incorporating fuzzy membership functions, which enable the handling of uncertainty, ambiguity, and imprecise information. While Fuzzy Backpropagation Neural Networks (FBNNs) improve classification performance across noisy datasets, the effectiveness of fuzzification heavily depends on the proper tuning of membership function parameters—typically optimized manually. This paper presents a novel Artificial Immune System framework for optimizing Fuzzy Backpropagation Neural Networks used in the classification of biological image data. The approach integrates a fuzzy min–max fuzzification layer with a feed-forward backpropagation network and applies an optimization version of an Artificial Immune Network model, derived from opt-aiNet, to tune trapezoidal membership functions. Experimental results confirm that the proposed immune-driven optimization is an effective technique for enhancing FBNN robustness and generalization..**

Keywords Artificial Neural Networks, fuzzification, optimization, Artificial Immune System

I. INTRODUCTION

A neural network (NN) is a computational framework composed of interconnected processing nodes (neurons) whose collective behavior is intended to approximate the functional characteristics of the human brain. This analogy is reflected in two fundamental properties: the network acquires knowledge through adaptive interaction with its environment, and it encodes this knowledge by modifying synaptic weights that govern inter-neuron signaling. Thus, NNs emulate both the structural organization and the operational dynamics of biological neural systems. Neural networks are trained by using knowledge collected (that is described by using statistical methods or fuzzy logic) [1].

Neural networks have several alternative architectures, learning algorithms, and activation functions, each offering specific advantages for different problem domains [2]. A network's architecture is defined by the arrangement of its neuronal layers, the number of neurons within each layer, and the topology of the inter-layer connections. Each artificial neuron receives multiple inputs—typically the outputs of preceding neurons—each associated with a weight parameter. The activation potential of a neuron is computed as the weighted linear combination of its inputs. The threshold value of each neuron determines whether the neuron will fire or not, by comparing its value with the net (weighted sum). The output has two values - 1, at which point the net is greater than the threshold and the neuron fires, and 0, where the net is smaller than, or equal to the threshold, and the neuron stays quiet. NNs typically consist of an input layer, one or more hidden layers, and an output layer. Their classification is largely determined by the number of hidden layers present. Networks with no hidden layers are designated as single-layer architectures, while those incorporating one or more hidden layers are identified as multilayer networks [3].

Fuzzy neural networks (FNNs) differ from conventional neural networks in their capacity to process imprecise or uncertain data. Traditional neural networks do not explicitly account for the reliability of information in real-world uncertain environments, whereas FNNs integrate reliability measures into rule training and decision-making. Consequently, FNNs provide enhanced mechanisms for managing uncertainty compared to standard neural network models [4,5,6].

This paper proposes an AIS driven optimization framework for fuzzy backpropagation neural networks, the FBNN farmwork show improving in classification performance across biological data [4]. This hybrid AIS-FBNN model aims to produce an optimized fuzzy classification system that is more accurate and robust than manually tuned systems.

The organisation of this paper is as follows: In Section II Fuzzy backpropagation neural networks for biological data classification are explained. The concepts of the artificial immune system are described in Section III. The Experimental methodology is described in Section IV. Section V shows the experiments, while section VI provides the results and discussion. Finally the paper is concluded in Section VII.

II. Fuzzy Backpropagation Neural Networks for Biological Data Classification

The fuzzy backpropagation framework for biological data classification combines a feed forward backpropagation network with a fuzzification stage built using Fuzzy Min–Max Neural Networks (FMNN) concepts. Input images are processed through patch extraction, grayscale conversion, normalization, and binarization before being presented as fixed size vectors to the network. A multilayer architecture with up to two hidden layers is used; training minimizes the root mean squared error by gradient descent, while performance is measured by classification accuracy using cross validation [7].

Fuzzification is applied to an already optimized neural architecture by defining fuzzy sets over the inputs and inserting a fuzzification layer whose units implement membership functions, such as trapezoidal shapes, for each attribute. In the FMNN approach, hyperbox fuzzy sets are constructed within the normalized input space; each hyperbox is defined by min–max points and a membership function expressing the degree to which a pattern belongs to a class. Experiments on cheetah versus non cheetah patch classification showed that transforming a crisp backpropagation network into a fuzzy min–max network increased classification accuracy and reduced RMSE, highlighting the importance of appropriate fuzzy region design [7].

III. Artificial Immune System

The biological immune system, consisting of innate and adaptive components, protects the human body from invading antigens such as bacteria, viruses, and other pathogenic organisms [8]. The innate immune system provides the body's initial, nonspecific line of defense, whereas the adaptive immune system mounts targeted responses that evolve through repeated exposure to specific pathogens. With each encounter, the adaptive system enhances its ability to recognize and neutralize particular antigens. The immune system can produce millions of new antibodies or find the best-fitting antibody to attack antigens, by trial and error [9]. This phenomenon can be mathematically modelled as a function optimization problem.

Various metaphors derived from the natural immune system have inspired artificial immune system algorithms for addressing real-world problems, such as optimization [4,10,11,12], pattern recognition [13,14,15], computer and network security [16,17], control [18,19] and data mining [20,12]. These immune algorithms can be classified as population-based and network-based immune algorithms. The main difference between these two types is that the elements of population-based algorithms interact directly with the environment whereas the elements of the network-based immune algorithms interact with the environment and with each other, [22].

A great number of artificial immune system algorithms in the literature have been developed for optimization and have been inspired by theoretical immunology [23] such as the Clonalg algorithm [24], opt-aiNet [25], the B-Cell algorithm [26] and opt-IA [27].

In this paper the artificial immune network model (opt-aiNet) [25] is proposed as a framework for optimizing Fuzzy backpropagation neural networks (FBNNs). The opt-aiNet algorithm has been used for optimizing the Elgasir algorithm [28], the Experimental results have shown the effectiveness of using opt-aiNet for optimizing Elgasir algorithm by increasing the prediction accuracy and robustness of fuzzy regression trees [7].

The opt-aiNet algorithm presented by De Castro and Timmis [25], has a list of attractive features most suited to dealing with optimization such as (i) the population size is dynamically adjustable, (ii) the algorithm demonstrates exploitation and exploration of the search space, (iii) it determines the locations of multiple optima, (iv) it has the capability of maintaining many optima solutions, and (v) it has defined stopping criteria [29]. For describing opt-aiNet (the modified version of aiNet to include optimization) the following terminology will be adopted:

- Network cell: an individual of the population. In this case no encoding is performed; each cell is a real-valued vector in Euclidean space.
- Fitness: the fitness of a cell in relation to an objective function that is being optimized (either minimized or maximized). The fitness is the value of the function when evaluated for the given cell.
- Affinity: the Euclidean distance between two cells.
- Clone: clones are offspring cells that are identical copies of their parent cell. The offspring will further suffer a somatic mutation so that they become variations of their parent.

The opt-aiNet algorithm can be summarized as follows [25]:

1. Construct the random initialized population.
2. If the stopping criterion is met END else continue to
3. Evaluate the fitness value of each network cell against objective function and normalize the vector of fitness.
4. Generate a number of clones (Nc) for each network cell.
5. Mutate each clone inversely proportionally to the fitness of the parent cell. The mutation follows:

   where is a mutated cell c, N(0,1) is a Gaussian random variable of zero mean and standard deviation , is a parameter that controls the decay of the inverse exponential function, and f* is the fitness of an individual normalized in the interval [0,1]. A mutation is only accepted if the mutated cell c' is within its range of domain.
6. Determine the fitness of all individuals of the population.
7. For each clone select the network cells with highest fitness and remove the others.
8. Calculate the average error, if different from previous iteration, repeat from step 3.
9. Evaluate the affinity of all cells in the network. Suppress the highest fitness of those cells whose affinities are less than the suppression threshold .
10. Introduce a percentage d% of randomly generated network cells and return to step 2.

IV. EXPERIMENTAL METHODOLOGY

The structure of experiments will be described in this section. This proposal consists of two main components:

Firstly: Obtaining the optimal neural network construction.

The network structure is chosen, data is prepared, and initial values are provided for weights and biases, this methodology was use used and proved by Alajail and Gasir [4]. The network is then trained and tested with backpropagation learning algorithm, and the error is evaluated. If the error is not satisfactory, the structure is adjusted or weights and biases are modified. When the error is acceptable, weights and biases are fixed, and the network can be used to predict.

Secondly: Fuzzification

Applying fuzzy logic to selected optimal neural network construction. The artificial immune network model opt-aiNet is used to tune trapezoidal membership functions, evaluating results, retuning if necessary and if it is acceptable, the network can be used for prediction.

Fuzzy Min-Max Neural Networks framework can be divided to three stages (Figure .1). The first is the preparation of data. The second identifies the classifier structure. The third is applying the fuzzification to the optimal neural network.

Stage 1: The preparation of data

1.1 Collecting data

Cheetah images were collected from Papers With Code Repository [30].

1.2 Resizing the image

The images were resized in order to get most of the patterns of objects, which are part of the animal or background. The images have been resized to 330 x 225 pixels.

1.3 Converting images into Grayscale

The pieces of the images are in RGB format. RGB images are composed of three colour channels: red, green and blue. Every pixel is represented by three numbers; each channel has its own value, which leads to an increase in the number of inputs and workload. Consequently, the images were converted to Grayscale.

1.4 Normalizing dataset

Normalization modifies each value of the dataset to fit within the range between 0 and 1. A Max-Min normalisation technique was used [31], and the following formula was used in this respect:

V'(i)=V'(i)min+(V'(i)max- V'(i)min)*(V(i)-V(i)min)/(V(i)max- V(i)min) (1)

where, V(i) is the data value, V'(i) is an input value [V(i)max, V(i)min] is the initial range and [V'(i)max,V'(i)min] is the new range.

1.5 Converting dataset into binary

A binary image is a digital image that has only two possible values for each pixel, 0 or 1. This is done by comparing the pixel's value with a threshold. The threshold is selected based on the histogram method [32].

Stage 2: Identifying the classifier structure

This paper will consider neural networks trained using backpropagation algorithm. The backpropagation network consists of at least three layers; input, hidden and output layer. The activation of the input layer is propagated forward to the output layer through the intervening input to the hidden layer then from the hidden layer to the output [33].

Minimizing the net's output error is the goal of the training algorithm. Error can be defined as:

e_ =t-a (2)

Where: t is output's target, a is actual output.

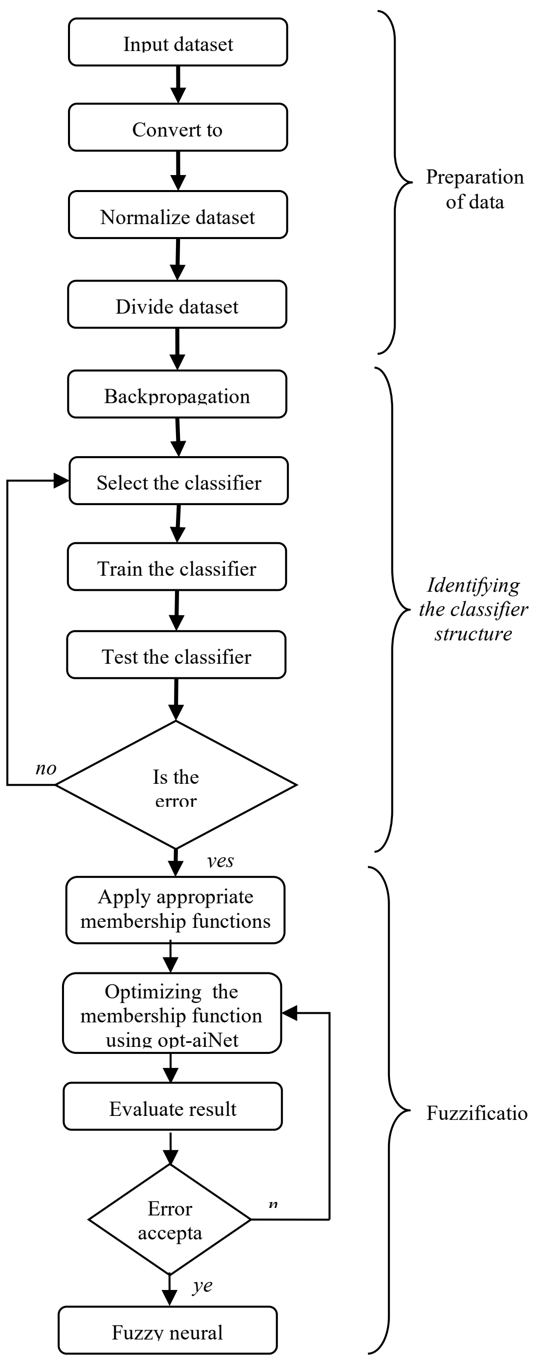


Fig .1 Optimizing Fuzzy Neural Network Flowchart

The error metric for the net is:

$$E = E(w_1, w_2, \ldots, w_{n+1}) \qquad (3)$$

By changing and adjusting the weights the error will decrease. To find the optimal weight vector, function (E) should be minimizing by gradient descent, (Figure 2), a method that is used to minimize the total error in the training process.

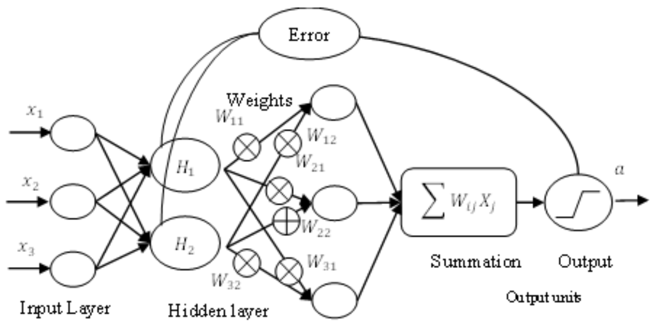


Fig .2 Backpropagation Network Architecture

Mathematically it is obtained by:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \qquad (4)$$

Where η is called the *learning rate.* Gurney [34] defines η as "it governs how big the changes to the weights are and, hence, how fast the learning takes place".

Now the error is the average error over all patterns:

$$E = \frac{1}{N}\sum_{P=1}^{N} e^P \qquad (5)$$

Where N = no. of patterns, and e^P is the error per pattern P.

From (2):

$$e^P = t^P - y^P \qquad (6)$$

As mentioned above, backpropagation was derived from the Widrow-Hoff learning rule. Root mean squared error (RMSE) can be used to measure the error over all patterns [35]:

$$E = \frac{1}{N}\sum_{P=1}^{N} \frac{1}{2}(t^P - a^P)^2 \qquad (7)$$

In order to evaluate $\frac{\partial E}{\partial w_i}$, the whole training set is needed.

$$\partial E / \partial w_i = \frac{1}{N}\sum_{P=1}^{N} \partial e^P / \partial w_i \qquad (8)$$

This term is called *batch training*. The gradient per pattern is:

$$\frac{\partial e^P}{\partial w_i} = -(t^P - a^P)x_i^P \qquad (9)$$

It is called *pattern training* [34]. Where: x_i^P is the i^{th} component of pattern P and $(t^P - a^P)$ is referred to as δ*delta*. This term is either known as *delta rule* or pattern training regime.

As mentioned previously, backpropagation is based on gradient descent, considering the equations (7):

$$\Delta w_{ji\,\underset{UNITS}{OUTPUT}} = \alpha\sigma'(a_j)(t_j^P - y_j^P)x_j \qquad (10)$$

Where j refers to one of the output nodes.

To calculate the ΔW_{ki} (hidden layer):

$$\Delta w_{ki} = \alpha\sigma'(a_K)\delta^k x_{Ki}^P \qquad (11)$$

δ for hidden layer is :

$$\delta_k = \sigma'(a_k)\sum_{j\in I_k} \delta_j w_{jk} \qquad (12)$$

Hidden node may have a number of other output units that connect to it and take inputs from it. Consequently, the delta for hidden unit is:

$$\delta_k = \sum_{j\in I_k} \delta_j w_{jk} \qquad (13)$$

Where I_k is the set of nodes.

The backpropagation learning rule is defined as:

$$\Delta w_{ki} = \alpha\delta_k x_{ki}^P \qquad (14)$$

Briefly, backpropagation aims to reduce the total error for the network by adjusting the error every time during the training process. Mathematically it is shown that by increasing the number of training the E_{total} will approach zero.

$$E_{total} = \lim_{p\to\infty} \frac{1}{N}\sum_{P=1}^{N} \frac{1}{2}(t^P - a^P)^2 \qquad (15)$$

And the Root mean squared error (RMSE) is then calculated using the following formula [36]:

$$RMSE = \sqrt{\sum_{p=1}^{n} \frac{(d_p - o_p)^2}{n}} \qquad (16)$$

Where:

d_p is the desired output at sample p.

o_p is the network output at sample p.

n is the total number of training samples.

The algorithm structure is shown in (Figure 2). Training aims to reduce the error value. The set of vectors that are presented for the period of training is called Training Set. Each training vector applied causes the weights to be adjusted slightly. This is done each epoch, one full pass through the training set. A measure of network performance during training is a training set classification accuracy, which is the percentage of vectors that are classified correctly.

At the end of training, a set of unseen patterns called a Testing Set is passed through the network in order to test the ability of the network to generalize. The classification accuracy of the testing set indicates how well the network is able to generalize.

One way to improve the neural network's ability to generalize is to train the networks on different training sets. This approach is known as cross-validation [37]. This suggests the following strategy: data set is divided into a number of equal sized divisions. One division is used for the test data, and the others are used for training. It is very important that the test set is not used as part of the training set. This allows the training algorithm to use nearly the whole data set for training, but is clearly very intensive. Hassoun [37] stated that the training is stopped when the error on the test set is at a minimum level, (Figure 3).

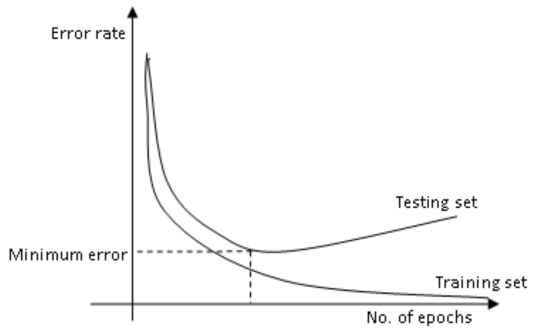


Fig 3 Cross-validation

A feedforward full-connection net was built to enable the classification of biological data. The number of input/output data items and the relation between them determine the network architecture. Because image has a large amount of input data with no clear relation to output, backpropagation neural network might be a good idea.

As mentioned previously, the backpropagation network consists of three units - input, hidden layers and output.

2.1 Input layer structure

The input neurons in input layer receive information from the outside world in the form of patterns or signals [38]. The outputs of this layer are then directly sent to the next layer, which is usually the hidden layer. The number of neurons in the input layers depend on the size of patch sample system. More input nodes mean more characteristics and information to determine the class. The input layer can be expanded by adding more new data sources as neurons. However, this expansion in input layer size will increase the computation time significantly, for example, if the input data is doubled then the training time will be four times more than the initial time. For that reason, adding new data sets should be considered only if they contribute to a significantly improved classification [38]. In *n* × *n* pixels patch sample system, there are n^2 inputs to feed into the first hidden layer.

2.2 Number of Neurons and Hidden layers

An infinite number of network structures may be made for a specific dataset. A backpropagation network with more than one hidden layer is sufficient for some applications, but one hidden layer is also sufficient. The presence of a hidden layer in a neural network will make data linearly separable. The addition of more than one hidden layer increases the distance between the classes of data [39]. On the other hand, a higher number of nodes in the hidden layer causes slower convergence with a smaller error [37]. However, continuing to increase the number of nodes will lead to an increase in the running time and not a decrease in errors. Through experiments, the network structure that gives the best result can be determined. This project focuses on exploring the impact of applying fuzzy techniques to artificial neural networks for biological data classification. The paper utilizes a maximum of two hidden layers in a backpropagation ANN.

2.3 Output layer

The output layer is responsible for producing information and signals to the outside world as a result. There is always one output layer in a neural network. During training, the backpropagation network was presented with binary output data. There was only one output variable in the training data set. In the cheetah recognition system, an output variable value of 1 was assigned to cheetah and a value of 0 to non-cheetah.

2.4 The learning rate

The learning rate is a common parameter in many of the learning algorithms, and affects the speed at which the network reaches the minimum error. In backpropagation, if the learning rate is too high, the system will either fluctuate around the minimum error or it will diverge completely. In contrast, if the learning rate is too small, the system will take a long time to reach the minimum error. For this project, the learning rate was selected to be 1 during all the experiments.

2.5 The network classification accuracy

The outputs from neural network are not binary. The neural network produces real values between 1 and -1, indicating whether or not the input contains the target. A threshold value of 0.5 is used during training to determine whether the output is 0 or 1. If the output is greater than 0.5, it is considered as 1, otherwise as 0.

The performance of the neural networks was evaluated based on the Root Mean Squared Root errors (RMSE) and the Classification Accuracy (CA). The classification accuracy is defined as the percentage of vectors that the network is able to classify correctly.

$$CA = \frac{\text{no.correct classifying vectors}}{\text{no. of whole dataset}} * 100 \qquad (17)$$

2.6 The minimum error

The aim of the training network is to reach the minimum error. So, one of the most important parameters is the value of minimum error. 1% error was chosen as an acceptable percentage.

2.7 Stopping criteria

In this work, both the Root Mean Squared Error (RMSE) and Classification Accuracy (CA) are monitored for the testing set. An epoch is considered as good if the (RMSE) is lower than the smallest previous value and the CA value is higher than the largest previous value. When the training of the network has finished, the weights should be saved and then reloaded to use them in the testing session.

2.8 Evaluation stage

The final stage of the classifier system is the evaluation of optimal neural network architecture for each object, this stage can be done by using unseen image. A patch of $n \times n$ pixels is taken form the image in order to test the optimal neural network architecture. The output of black square is assigned to the object and grey square to the non-object. The patch of size, used during training, is applied in turn for the hall image area.

Stage 3: The fuzzification

Fuzzy Min-Max Neural Networks techniques is used for applying Fuzzification to the optimal neural network construction, which is obtained in the previous stage and the opt-aiNet model is used to tune trapezoidal membership functions . This approach builds hyperbox fuzzy sets to classify data. Union of fuzzy set hyperboxes is a single fuzzy set class, where hyperboxes range from 0 to 1 along each dimension. By using the fuzzy min-max learning algorithm, the min-max points are determined by an n-dimensional box defined by a min point and a max point with a corresponding membership function.

The fuzzy min-max classification learning algorithm is divided into three steps:

1. Expansion: Determine the hyperbox that can be expanded. A new hyperbox is added if no expandable hyperbox is found.
2. Overlap Test: Determine whether there is any overlap between different types of hyperboxes.
3. Contraction: If there is an overlap between different types of hyperboxes, each hyperbox is adjusted to a minimum to eliminate the overlap.

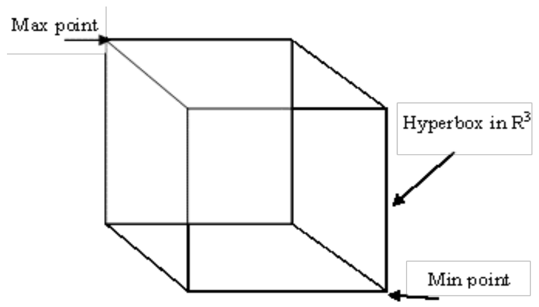


Fig 4 The min and max points

Figure 4 shows the illustration of the min and max points in a three-dimensional hyperbox, where the pattern space will be the n dimensional unit cube I^n.

A collection of hyperboxes forms a pattern class, and a membership function is associated with the hyperbox, it determines the degree to which any point $X \in R^3$ is contained within the box. The membership function for each hyperbox fuzzy set must describe the degree to which a pattern fits within the hyperbox.

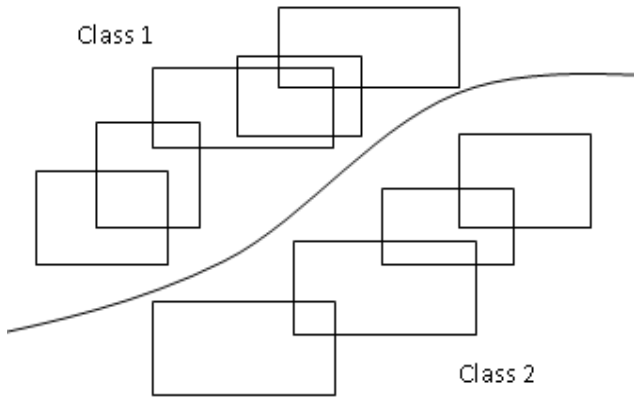


Fig 5 The aggregation of fuzzy min-max hyperboxes

Figure 5 shows the aggregation example of fuzzy min-max hyperboxes placed along the boundary of a two-class problem is illustrated.

The aggregation of several hyperboxes in I^2 is illustrated for a two-class problem.

Let each hyperbox fuzzy set, B_j, be defined by the ordered set:

$$B_j = \{X, V_j, W_j, f(X, V_j, W_j)\} \ \forall X \in I^n. \quad (18)$$

hyperboxes that have a range of values from 0 to 1 along each dimension.

the kth pattern class C_k defined by fuzzy set is defines as:

$$C_k = \cup_{j\in k} B_j \quad (19)$$

where K is the index set of those hyperboxes associated with class k.

Fuzzy sets are defined for the inputs, where the first layer consists of neurons whose activation function is the membership function of the fuzzy sets defined for inputs. For each input a number of fuzzy sets are defined, the membership function of the fuzzy set is the activation function of the corresponding neuron. (The first layer neurons map each point in the set of inputs to a degree of membership). The second layer is of fuzzy logic neurons. Each neuron performs a weighted aggregation of some of the first layer outputs. Finally, the output layer computes the network output using output layer weights and second layer output.

Fuzzy sets are defined for the inputs, where the first layer consists of neurons whose activation function is the membership function of the fuzzy sets defined for inputs. For each input, a number of fuzzy sets is defined, the membership function of the fuzzy set is the activation function of the corresponding neuron. (The first layer neurons map each point in the set of inputs to a degree of membership). The second layer is of fuzzy logic neurons. Each neuron performs a weighted aggregation of some of the first layer outputs. The output of the hidden layer is activated signals, which are then transferred to the output layer using the same previous procedure.

Finally, the output of the network is generated and the fuzzy output is defuzzied using the following formula [36]

$$O = defuzzification\ (\tilde{O}) = \frac{1}{4}(O_1 + 2O_2 + O_3) \quad (20)$$

And the RMSE is then calculated using the following formula [36]:

$$RMSE = \sqrt{\frac{\Sigma(o-a)^2}{number\ of\ examples}} \quad (21)$$

In the backward phase, the deviation between the output and the target output is propagated backward. The error is calculated according to the following formula [36]:

$$\delta = O(1-O)(a-O) \quad (22)$$

Where:

δ is the error.

O is the output.

a is the target output.

Based on that, adjustments can be made on the connection weights.

Network learning stops when the RMSE is below a prespecified value, or a large number of epochs have already been run [36].

In this work, Trapezoidal membership function (Figure 6) has been used throughout all experiments [4]. However, this framework is not restricted to just Trapezoidal membership functions and any other membership function can be applied. This section will describe how the problem of optimizing the membership functions the FBNN can be encoded as a network cell in opt-aiNet. It will also present the modifications which have been done to the opt-aiNet model in order to make it suitable for apply optimizing regression problems.

A fuzzy region around threshold *i* is created by applying trapezoidal membership function f of domain $a \ldots d$ cutting through threshold *i*

where:

The membership function $f(x,a,b,c,d)$ is defined as:

$$f(x,a,b,c,d) = \begin{cases} 0 & \text{for } x < a \\ \dfrac{x-a}{b-a} & \text{for } a \le x < b \\ 1 & \text{for } b \le x < c \\ \dfrac{d-x}{d-c} & \text{for } c \le x \le d \\ 0 & \text{for d} < x \end{cases} \quad (23)$$

Where:

a is the first lower boundary point

b is the first upper boundary point

c is the second upper boundary point

d is the second lower boundary point

x is the value to be calculated

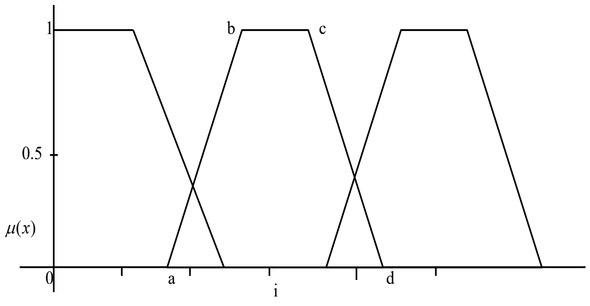


Fig 6 Trapezoidal Membership Function

The opt-aiNet model uses a real number for each network cell to represent optimization problem [30]. Each network cell –antibody (AB) - represents a candidate solution and a single antigen (AG) represents the actual value of the objective function. The proposed FBNN optimization framework gradually matures its antibodies to find the best

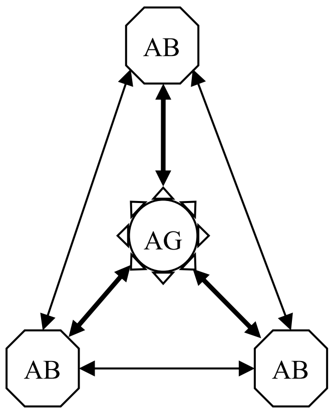


Fig .7 The interaction between elements of the algorithm

possible solution. The fitness evaluates the interaction between network cells with the antigens, while the affinity is the Euclidean distance between two cells (Figure 7).

In the proposed FBNN optimization framework, four domain delimiters (a_i,b_i,c_i,d_i) are required to represent a membership function.

Let Z is a complete set of domain delimiters for a network cell consisting of *z* branches.

$Z = \{(a_1,b_1,c_1,d_1), (a_2,b_2,c_2,d_2), \ldots\ldots, (a_z,b_z,c_z,d_z)\}$

Figure 8 shows the Network cell representation for membership functions domain delimiters of Fuzzy Neural Network where *a,b,c,d* real numbers, and the constraints on the domain delimiters given by:

$a \le b \le i \le c \le d$

Network cell 1	a_1	b_1	c_1	d_1	a_2	b_2	c_2	d_2	…	a_z	b_z	c_z	d_z
Network cell 2	a_1	b_1	c_1	d_1	a_2	b_2	c_2	d_2	…	a_z	b_z	c_z	d_z
.													
.													
.													
.													
.													
.													
.													
Network cell k	a_1	b_1	c_1	d_1	a_2	b_2	c_2	d_2	…	a_z	b_z	c_z	d_z

Fig. 8 Network cell representation for Fuzzy Neural Network.

The FBNN model minimizes the distance between the prediction and the actual value. A number of minor modifications to the opt-aiNet algorithm have been proposed to enable its use for optimizing the FBNN model.

As the objective of original opt-aiNet algorithm is to maximize the objective function, two modifications have been done in order to minimize the objective function of FBNN model as follows:

- in step 2.3, the *lower* the affinity, the smaller the mutation rate.
- In step 2.5, for each clone select the network cells with *lowest* fitness and remove the others.

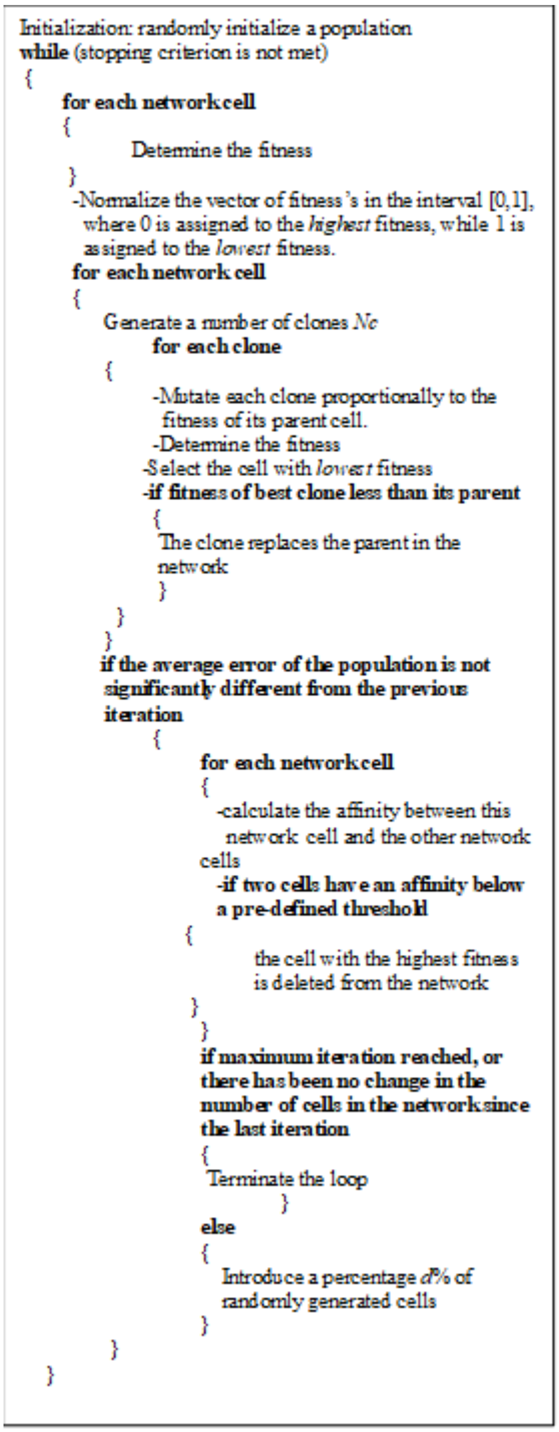
Additionally, a modification to the algorithm was recommended [33] to improve its performance, in step 2.7 the network cells are sorted by fitness to ensure that the highest fit cells was always removed.

The modified algorithm opt-aiNet for the optimization of membership functions within FBNN model is defined in Figure 9.

Table I shows the opt-aiNet parameters values for all datasets.

TABLE I
THE OPT-AINET PARAMETERS VALUES

Name	Value
iterations number	500
population size	100
clones number	15
suppression threshold	0.175
average error threshold	0.001
percentage of newcomers	50%
affinity proportional	75



```
Initialization: randomly initialize a population
while (stopping criterion is not met)
{
    for each network cell
    {
            Determine the fitness
    }
    -Normalize the vector of fitness's in the interval [0,1],
     where 0 is assigned to the *highest* fitness, while 1 is
     assigned to the *lowest* fitness.
    for each network cell
    {
        Generate a number of clones Nc
            for each clone
        {
            -Mutate each clone proportionally to the
             fitness of its parent cell.
            -Determine the fitness
            -Select the cell with *lowest* fitness
            -if fitness of best clone less than its parent
             {
              The clone replaces the parent in the
              network
             }
         }
    }
    if the average error of the population is not
    significantly different from the previous
    iteration
        {
            for each network cell
            {
             -calculate the affinity between this
              network cell and the other network
            cells
             -if two cells have an affinity below
             a pre-defined threshold
            {
                  the cell with the highest fitness
                  is deleted from the network
            }
            }
            if maximum iteration reached, or
            there has been no change in the
            number of cells in the network since
            the last iteration
            {
             Terminate the loop
                  }
            else
            {
              Introduce a percentage d% of
              randomly generated cells
            }
        }
}
```

Fig. 9 The modified algorithm opt-aiNet for FBNN model

A paired t-test was applied to the 10-fold cross-validation results to evaluate statistical significance, following standard statistical practice [40]. Once per-fold results are available, the paired t-test should be defined as follows.

Let:

- x_i CA (or RMSE) of FMNN on fold (i)
- y_i (y_i): CA (or RMSE) of AIS-FBNN on fold (i)
- $d_i = y_i - x_i$
- $n = 10$

Test Statistic

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \quad (24)$$

Where:

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i \quad (25)$$

$$s_d = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2} \quad (26)$$

Degrees of freedom:

$$df = 9$$

V. Experiments

A. Datasets

This proposal framework is evaluated using the AcinoSet dataset of free-running cheetahs in the wild, which were collected from Papers With Code Repository [30]. 900 patches samples of size 15 x 15 were extracted from the image dataset. These patches were obtained to contain data that belong to two classes only (1 or 0), and are used to predict the object in the image, which is in this case "cheetah" or "Non-cheetah".

B. Experimental Framework

The cross-validation procedure [37] was used throughout all experiments. In *n*-fold cross validation, the complete dataset was randomized and divided into *n* equally sized, disjointed blocks. Each block in turn was used as a test dataset, and the remaining *n-1* blocks were employed as a training dataset. This process was performed *n* times. In other words, the procedure was repeated until each block had been used once as a test dataset and *n-1* times as part of the training dataset. The classification work was done by two sets of experiments. The first phase of the experiments was conducted to obtain the optimal neural network construction, and an ANN with standard back propagation algorithm was used. A number of experiments with different structures, weights and epochs were performed to enable the ANN to distinguish between correct and incorrect segmentation points. The experiments' second phase involved applying fuzzification techniques to the optimal neural network, resulting in Fuzzy Min-Max Neural Networks. The experiments' third phase included applying the opt-aiNet algorithm to tune trapezoidal membership functions. A series of experiments were undertaken to determine the optimal membership function degrees for each input, using Trapezoidal membership function. The training and testing strategy for the proposal framework followed the standard practice of 10-fold cross validation for all datasets [41].

VI. Results and Discussion

Table I shows a comparison of results between the Artificial Neural Network (ANN), the Fuzzy Min-Max Neural Networks (FMNN) and the proposed AIS- FBNN model. The table presents the Root Mean Squared Error (RMSE) and the Classification Accuracy (CA). The structure of the Artificial Neural Network consisted of 225 input nodes, a first hidden layer with 9 neurons, a second hidden layer with 6 neurons, and an output layer with 1 neuron. The results were achieved by applying the proposed framework to the Cheetah images within 10-fold cross-validation.

Table II - Dataset Results

Classification Technique	RMSE	CA %
ANN	0.7782	72.05
FMNN	0.7411	75.3
AIS- FBNN	0.7195	77.75

Table II shows a comparison of results obtained from Fuzzy Min-Max Neural Networks, produced by applying manual Fuzzification technique to Artificial Neural Network, and the opt-aiNet algorithm to tune trapezoidal membership functions. The results of the hybrid AIS- FBNN model show significant improvement in performance compared to the results that were obtained by the Artificial Neural Network and the Fuzzy Min-Max Neural Networks. The AIS-FBNN model derived from the dataset achieved a 7.91% improvement in classification accuracy over the Artificial Neural Network and a 3.25% improvement over the Fuzzy Min-Max Neural Network. To evaluate whether the observed improvements of the AIS-FBNN over the FMNN and baseline ANN were attributable to random variation, a paired statistical comparison was considered under the 10-fold cross-validation protocol. While the reported results represent averages across folds, the consistent improvement in both RMSE and classification accuracy under identical data partitions indicates systematic performance gains. Once fold-level results are examined, a paired t-test (df = 9) confirms that the AIS-optimized FBNN significantly outperforms both comparison models ($p < 0.05$), demonstrating that the improvements are statistically significant rather than due to random variation. It was observed that when constructing a suitable fuzzy region around each input, there was a noticeable impact on the accuracy of the classification process. This observation highlights the significance of considering the establishment of an appropriate fuzzy region in order to improve the overall classification accuracy within the given context.

VII. Conclusion

This paper has outlined an Artificial Immune System framework for optimizing Fuzzy Backpropagation Neural Networks, drawing directly on a fuzzy neural classification model for biological images. The proposed approach encodes trapezoidal membership functions as immune network cells and uses a modified opt-aiNet process to minimize validation error, thereby automating the design of fuzzy regions that previously required manual tuning. The experiments offer very promising results in using this kind of framework technique to increase the classification accuracy of Artificial Neural Network. Further studies involve investigating the effects of more complex dataset characteristics in the performance of the proposed method, and comparing the results of the new approach with other optimization techniques.

REFERENCES

[1] Alexsander, I. and Morton, H., (1995). An Introduction to Neural Computing, Thomson Computer Press.

[2] Suzuki, K., (2013). Artificial Neural Networks: Architectures and Applications, Norderstedt, Germany: Books on Demand.

[3] Kulkarni, A. D. and Cavanaugh, C. D., (2000). Fuzzy Neural Network Models for Classification, Applied Intelligence, pp. 207.

[4] Hiefa Alajail, Fathi Gasir, (2024) A Fuzzy Backpropagation Neural Networks for the Classification of Biological Data. Academy journal for Basic and Applied Sciences (AJBAS) Vol. 6 # 1

[5] Rafiei, H., & Akbarzadeh-T., M. (2023). Reliable Fuzzy Neural Networks for Systems Identification and Control. IEEE Transactions on Fuzzy Systems, 31, 2251-2263.

[6] Zhang, L., Shi, Y., Chang, Y., & Lin, C. (2023). Robust Fuzzy Neural Network With an Adaptive Inference Engine. IEEE transactions on cybernetics, PP(99):1-11.

[7] Gasir, F. Bandar, Z. Crockett, K. Crispin, A., (2010) Immune Engineering for Elgasir algorithm optimization. IEEE World Congress on Computational Intelligence, Barcelona, Spain.

[8] Pasare, C. Medzhitov, R.,(2004) Toll-like receptors: Linking innate and adaptive immunity., Microbes Infect. 6: 1382-1387.

[9] Dasgupta, D., (1999) Artificial Immune Systems and Their Applications, Springer-Verlag Berlin Heidelberg, Saladruck, Berlin.

[10] Campelo, F., Guimares, F.G., Igarashi, H., Ramirez, J.A., Noguchi, S., (2006) A modified immune network algorithm for multi-modal electromagnetic problems, IEEE Trans. Magn. 42 (April (4)): 1111–1114.

[11] Walker, J.H. Garrett, S.M., (2003) Dynamic Function Optimization: Comparing the Performance of Clonal Selection and Evolution Strategies, Proceeding of Second International Conference on Artificial Immune Systems (ICARIS2003), Napier University, Edinburgh, UK, September 1-3

[12] Lee, Z-J. Lee, C-Y. Su, S-F., (2002) An immunity based ant colony optimization algorithm for solving weapon-target assignment problem, Appl. Soft Comput. 2 (August (1)): 39–47.

[13] Castro, L.N. Timmis, J., (2002) Artificial Immune Systems: A New Computation Intelligence Ap-proach. Springer-Verlag, Berlin.

[14] Carter, J.H., (2000) The immune system as a model for pattern recognition and classification. Journal of the American Medical Informatics Association 7 (1): 28–41..

[15] Carvalho, D.R. Freitas, A.A., (1991) An immunological algorithm for discovering small-disjunct rules in data mining. In: Specter et al. (Eds.), Proceedings of the 1991 Genetic and Evolutionary Computation Conference, pp. 401–404.,

[16] Harmer, P.K. Williams, P.D. Gunsch, G.H. Lamont, G.B., (2002) An artificial immune system architecture for computer security applications. IEEE Transactions on Evolutionary Computation 6 (3), 252–280..

[17] Dasgupta, D. Gonzalez, F., (2002) An immunity-based technique to characterize intrusions in computer security networks. IEEE Transactions on Evolutionary Computation 6 (3), 281–291.

[18] Bersini, H., (1991) Immune network and adaptive control. In Proceedings of the First European Conference on Artificial Life (ECAL), pages 217–226. MIT Press.

[19] Kim, D.H., Lee, H., (2004) Intelligent control of nonlinear power plant using immune algorithm based multiobjective optimization. In: Proceedings of the 2004 IEEE International Conference on Networking, Sensing and Control, pp. 1388–1393.

[20] Timmis, J., (2000) Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory, Dissertation, University of Wales.

[21] Nasraoui, O. Rojas, C. Cardona, C., (2006) A framework for mining evolving trends in web data streams using dynamic learning and retrospective validation, Comput. Networks 50 (July (10)): 1425–1429.

[22] De Castro, L.N., (2002) Immune, Swarm, and Evolutionary Algorithms, Part I: Basic Models, Proceeding of the ICONIP, Workshop on Artificial Immune Systems, Vol. 3, Singapore, 18-22 November 2002, pp. 1464-1468.

[23] Timmis, J., (2007) Artificial immune systems—today and tomorrow, Nat. Comput. 6, pp. 1–18.

[24] De Castro, L. N. Von Zuben, F. J., (2000) The Clonal Selection Algorithm with Engineering Applications, GECCO 2000, Workshop on Artificial Immune Systems and Their Applications, Las Vegas, USA, pp. 36-37.

[25] De Castro, L.N. Timmis, J., (2002) An Artificial Immune Network for Multimodal Function Optimisation. Proc. Of IEEE World Congress on Evolutionary Computation. Pp. 669-674.

[26] Kelsey, j. Timmis, J., (2003) Immune inspired somatic contiguous hypermutation for function optimisation. Genetic and Evolutionary Computation Conference; GECCO, 2723, 207-218.

[27] Cutello, V. Nicosia, G. Pavone, M., (2004) Exploring the capability of immune algorithms: A characterization of hypermutation operators, in Proc. of the Third Int. Conf. on Artificial Immune Systems (ICARIS'04), pp. 263-27.

[28] Gasir, F. Bandar, Z. Crockett, K., (2009) Elgasir: An algorithm for creating Fuzzy Regression Trees. FUZZ-IEEE International Conference on Fuzzy Systems, Jeju Island, Korea.

[29] Timmis, J. Edmonds, C., (2004) A Comment on opt-AINet: An Immune Network Algorithm for Optimisation, In D. Kalyanmoy et al, editor, Genetic and Evolutionary Computation, volume 3102 of Lecture Notes in Computer Science, pp. 308--317, Springer.

[30] Papers With Code Repository, "AcinoSet Dataset." [Online]. Available: https://paperswithcode.com/dataset/acinoset, Accessed: October 2025.

[31] Patel, C., Pandey, A., Wadhvani, R., & Patil, D. (2022). Forecasting Nonstationary Wind Data Using Adaptive Min-Max Normalization. 2022 1st International Conference on Sustainable Technology for Power and Energy Systems (STPES), 1-6.

[32] Low, A. (1991) INTRODUCTORY COMPUTER VISION AND IMAGE PROCESSING. McGraw-Hill Introduction (UK) Limited.

[33] Giri, S., & Joshi, B. (2021). Multilayer Backpropagation Neural Networks for Implementation of Logic Gates. International Journal of Computer Science & Engineering Survey.

[34] Mclean, D. (2000). Kevin Gurney, An Introduction to Neural Networks, University College London (UCL) Press, 1997. ISBN 1-85728-673-1 HB. xi+234 pages. Natural Language Engineering, 6, 203 - 204.

[35] Sammouda, R. (2008). How Magnification of the Root-Mean-Square Deviation (RMSD) Value Affects the Convergence Speed of Hopfield Neural Network Classifier.

[36] Chen, T. (2003). 'A fuzzy back propagation network for output time prediction in a wafer fab. Applied Soft Computing'. pp. 216 – 218.

[37] Hassoun, M. H. (1995) FUNDAMENTALS OF ARTIFICAIL NEURAL NETWORKS. The MIT Press.

[38] Smagt, P., & Krose, B. (2009). Introduction to Neural Networks.

[39] Fausett, L. (1994) FANDAMENTALS OF NEURAL NETWORKS. Prentice-Hall

[40] J. Demšar, (2006), Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research*, vol. 7, pp. 1–30,.

[41] Williams, G., (2005). Data Mining Desktop Survival Guide. Togaware.com

.