Training of RFB neural networks using a full-genetic approach

Iulian-Constantin VIZITIU, Petrică CIOTÎRNAE, Teofil OROIAN, Adrian RADU
Florin POPESCU, Cristian AVRAM
Communications and Electronic Systems Department
Military Technical Academy
George Cosbuc Avenue 81-83, 5th District, Bucharest
ROMANIA
vic@mta.ro ciotirnae@mta.ro oroiant@mta.ro i_radu@mta.ro fdpopy@mta.ro cavram@mta.ro
http://www.mta.ro

Abstract: - The efficiency of pattern recognition (PR) systems using RBF neural networks to implement their recognition function, depends a lot by the training algorithms of these neural networks and especially, by the specific techniques (e.g., supervised, clustering techniques etc.) used for RBF center positioning. Having as starting point the basic property of genetic algorithms (GA) to represent global searching tools, a full-genetic approach to assure optimization both connectivity and neural weights of RBF networks is proposed. In order to confirm the broached theoretical aspects and based on real pattern recognition task, a comparative study (as performance level) with others standard RBF training methods and SART neural network is also indicated.

Key-Words: - RBF neural networks, clustering techniques, genetic algorithms

1 Introduction

It is well-known that, RBF neural networks have their origin in the solution of the multivariate interpolation problem [1], [2]. The RBF networks in their standard form have only one hidden layer (see Fig.1). Properly trained, they can approximate an arbitrary function $f : \mathrm{R}^n \to \mathrm{R}$ mapping:

$$f(x) \approx h(x) = w_0 + \sum_{i=1}^{m} w_i z_i(x), \qquad (1)$$

where $x \in \mathrm{R}^n$, $\{w_i\}_{i=\overline{1,m}}$ denotes the neural weights, w_0 is the bias and $z_i(x)$ represents the activation function (also known as radial basis function), which is given by equation:

$$z_i(x) = \Phi\left(\frac{\|x - t_i\|}{\sigma_i}\right), \qquad (2)$$

where $\|\cdot\|$ is the Euclidian norm, $t_i \in \mathrm{R}^n (i = \overline{1,m})$ is the *center* of RBF function, σ_i is its *width* and finally, $\Phi(\cdot)$ is a nonlinear function that monotonically decreases (or increases) as x moves away from t_i (usually, by gaussian, cubic, multiquadratic type etc.). Note also that, a RBF neural network with more than one output can be used in other specific applications [3].

Generally speaking, the training of RBF neural networks consists in two important stages, namely:
t1) determination of RBF parameters $\{t_i, \sigma_i\}_{i=\overline{1,m}}$ (also known as RBF center selection or mapping);
t2) using a proper (e.g., supervised etc.) training procedure, fitting of the neural weights to the output neural layer.

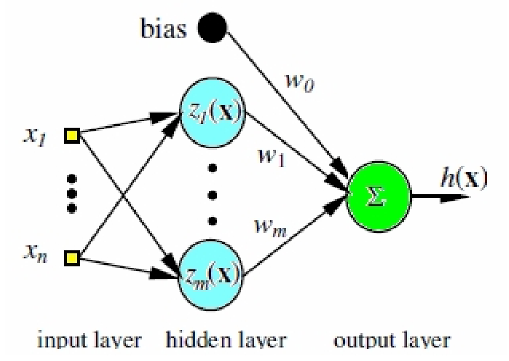


Fig.1: *n*-D RBF neural network architecture

According to special literature [2], [3], [4], the most important step in this training procedure is represented by the selection of RBF centers, and several basic mapping strategies have been proposed in this field: random positioning, supervised selection (e.g., Cholesky, LU or QR decompositions, SVD technique etc.), and *clustering* techniques (e.g., *k*-means algorithm, fuzzy-*c*-means, SOMs, OLS algorithm, ISODATA algorithm, RAN algorithm, UCC algorithm etc.), respectively.

As is expecting, a more efficient approach in RBF center mapping employs a *clustering* technique, such as for example, *k*-means, SOMs or other hybrid clustering algorithms [5], [6]. Let $\left\{(x_i, d_i)_{i=\overline{1,P}}\right\}$ be the set of the training examples, where x_i is an input vector and d_i its desired output. If at the ending of the applied clustering algorithm, the input vectors $\{x_i\}_{i=\overline{1,P}}$ were assigned to K clusters $S_1, S_2, ..., S_K$, then the mean vectors of each cluster given by equation:

$$m_i = \frac{1}{|S_i|}\sum_{x_i \in S_i} x_i, \tag{3}$$

become RBF centers (i.e., $t_i = m_i, i = \overline{1,K}$). Finally, in order to calculate the widths $\{\sigma_i\}_{i=\overline{1,K}}$, several inexpensive heuristic approaches are available. For example, in [4], it is suggested that a single value σ for all basis functions gives good results (in fact, it was used $\sigma = \left\langle \left\| t_i - t_j \right\| \right\rangle$, where t_j is the nearest center from t_i and $\langle \cdot \rangle$ indicates the average over all such pairs). Other methods use a different value σ_i for each basis function. In [7], each width was defined according to equation:

$$\sigma_i = \alpha \left\| t_i - t_j \right\|, \tag{4}$$

where α denotes an overlap factors.

Generally speaking, although the standard clustering algorithms (*k*-means, fuzzy-*c*-means, EM etc.) had been applied to many practical clustering problems successfully, it has been shown that these algorithms may fail to converge to a global minimum under certain conditions that represents certainly, an important disadvantage. Since *genetic algorithms* (GA) are powerful global searching tools, and are most appropriate for complex nonlinear models where location of the global solution is a difficult task, these algorithms can be applied with very good results, to evolve the proper number of clusters and to provide appropriate clustering, [5], [8], [9].

Accordingly, this new important class of hybrid clustering methods synthetically called *GA-based clustering algorithms*, could offer an excellent opportunity to be implemented inside of RBF neural network training algorithm, [6], [8], [10] (e.g., as RBF center mapping).

This mixed approach combines the robust nature of the genetic algorithms with the high-performance of the standard clustering methods. As a results, in the literature, a lot of GA-based clustering techniques and their different applications are described (e.g., GKA algorithm (and its faster version, FGKA algorithm), IGKA algorithm, GGA algorithm, GKMODE algorithm, GCUK algorithm, GCDC algorithm, GWKMA algorithm etc. [11], [12], [13], [14], [15], [16]).

This paper is aimed to present a *full*-genetic training technique of RBF neural networks having as starting point a suitable GA-based clustering method used as center positioning, and a genetic approach used in fitting of the output neural weights, respectively. All these proposed objectives will be checked against a real pattern recognition task. Therefore, in the first part of the paper, a theoretical description of genetic procedure used in training of RBF neural networks is described. Next, a comparative study of the proposed genetic method with others standard RBF training techniques and SART neural classifier is indicated. Finally, in the last part of the paper, the most important conclusions about the approached aspects, and some interesting future works in this field are also included.

2 Proposed GA approach

According to literature [2], [3], it is known that RBF networks represent as performance level, an efficient *alternative* to standard feedforward topologies (e.g., MLP neural networks), and the basic stages in their training process are represented by selection of RBF centers and fitting of the neural weights to the output layer, respectively. Generally speaking, the standard approaches used for RBF center positioning lead to some important *drawbacks* (e.g., different clustering methods can and do generate different solutions for the same dataset, improper behavior in case of very large datasets, problem of local minimums etc., [6]), but an optimal solution to increase the pattern classification performances of RBF networks could be represented by *genetic* selection of RBF centers [5], [8], [10]. Finally, to obtain a *full*-genetic training procedure of RBF networks, the standard supervised or unsupervised methods used ordinarily to calculate the neural weights from the hidden layer to the output layer, can also be replaced with a proper genetic optimization technique.

Accordingly, to obtain both center positioning and training rule of a RBF neural network, the proposed optimization procedure contained the following *two* processing modules:

m1) the task of the first processing module was to achieve the setting parameters of RBF $\{t_i, \sigma_i\}_{i=\overline{1,m}}$, where m represents the number of centers (or *hidden*

neurons). Accordingly, a GA-based clustering method containing the following basic steps is proposed:
s1) if the input training dataset is by the form $\{x_i, d_i\}_{i=\overline{1,P}}$, $x_i \in \mathrm{R}^n$ and K is the number (in this case, known) of the classes from the input space, then using an powerful (standard) clustering algorithm with *zoom* effect (e.g., *k*-means, fuzzy-*c*-means, ISODATA etc.), the most *natural* tendencies inside of each main data cluster were determined (i.e., each main data cluster was bounded into m_i new (sub)clusters where, $m = \sum_{i=1}^{K} m_i$). On the other hand, the basic rule to select the suitable clustering method was directly connected to the complexity (number of the clusters, cluster shapes etc.) of the input dataset. Finally, it can be observed that, using this first preliminary grouping procedure, a *pre*-clustering of the input space very useful for the next genetic optimization method, was thus obtained (see Fig.2);
s2) the starting chromosome population was made using a random selection of m_i vectors x_i from each class (i.e., one vector for each bounded cluster) and finally, a suitable linear concatenation. Therefore, each chromosome had assigned m vectors $\{t_i\}_{i=\overline{1,m}}$ which are extracted from the input dataset. To achieve a proper chromosomal representation, a *real* encoding technique was also used.

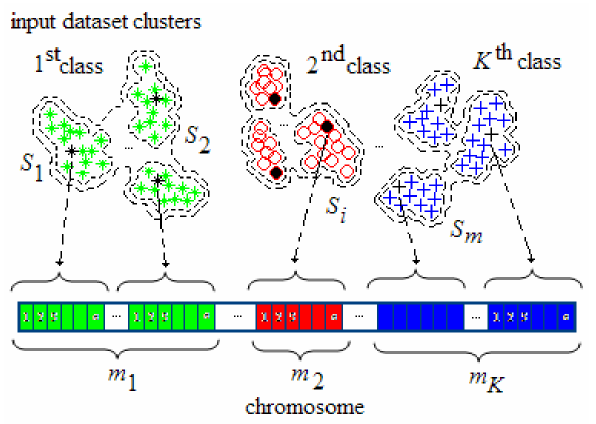


Fig.2: Genetic encoding technique used in case of RBF center positioning

The *fitness* used for each chromosome evaluation was calculated according to standard equation:

$$E = \left(1 + \left[\frac{1}{P}\sum_{i=1}^{P}(y_i - d_i)^2\right]^{0.5}\right)^{-1}, \qquad (5)$$

where y_i denotes the RBF network real output.

The *stopping criterion* was represented by the exceeding of the maxim generation number (this number has a constant value) or when the goal error was reached. The *selection* of the parents for the next chromosomal generation was also made using the well-known *roulette* principle, [9].

The continuous *crossover* supposed the use of two splitting points (randomly chosen), and each chromosome had attached a certain crossover probability with values into $[0.5, 0.85]$ range. To introduce new individuals inside of the current population and to protect GA against irreversible and accidental information failures generated by improper crossover operations, the uniform *mutation* operator was also used, [5].

To certainly obtain *the best* chromosomal solution offered by the proposed GA, a method similar with Gallant (or *pocket*) algorithm from neural network theory was used, [2].

Generally speaking, it is well-known that the solution given by GA is encoded under the form of the most performant chromosome belonging to the last chromosomal generation but in fact, nothing not garantees us that a more efficient chromosome was not already obtained, for example, in the previous chromosomal generation. Consequently, using the *natural* analogy with Gallant technique, at each chromosomal generation, the best individual from this population will be held into a virtual *pocket* and, after a suitable decreased order technique, certainly, the best GA solution will be thus obtained.

After the applying of RBF center selection procedure, $\{\sigma_i\}_{i=\overline{1,m}}$ width for each hidden neuron was calculated according to standard equation:

$$\sigma_i^2 = \frac{1}{P_i}\sum_{x_i \in S_i}(x_i - t_i)^T \cdot (x_i - t_i), i = \overline{1,m}, \qquad (6)$$

where S_i is the bounded cluster assigned to center t_i and P_i $\left(\sum_{i=1}^{m} P_i = P\right)$ is the number of training vectors x_i from this data subcluster.

Because in this moment RBF setting parameters $\{t_i, \sigma_i\}_{i=\overline{1,m}}$ are known, the neural weigths to the output layer $\{w_{ij}\}_{i=\overline{1,K}, j=\overline{1,m}}$ will be calculated using the second processing module which will be below described.

More details regarding the structure of GA used for RBF network center mapping can be found in [8].

m2) usually, a RBF neural network is trained using either standard supervised methods based in fact, on gradient descendent algorithm or unsupervised techniques (e.g., OLS algorithm etc.). Even though their popularity is indubitable, these training algorithms have some major ***drawbacks***, for example in case of supervised procedures, one of the most important being related to its *local* optimization character. Although some improved versions of these algorithms are indicated in literature [1], [2], an efficient solution to remove their basic disadvantages can be represented by the genetic algorithm use (e.g., it is known that, genetic algorithms represent powerful searching tools into complex data space, and which offer solutions by *global* type).

Having as starting point the RBF network parameters which was before determined (i.e., the best chromosomal solution offered by the first genetic module), the task of the second module was to optimize the distribution of the neural weights to the neural output layer. Consequently, for a suitable chromosomal representation of the neural weights, these were random initialized, and were ***real encoding*** into a linear structure so that, each chromosome represents a single output weights set (see Fig.3). Because the topology is predefined and remains fixedly after initialization of the training process, a single chromosome will contain only the values of neural weights and do not incorporate any information about its connectivity.

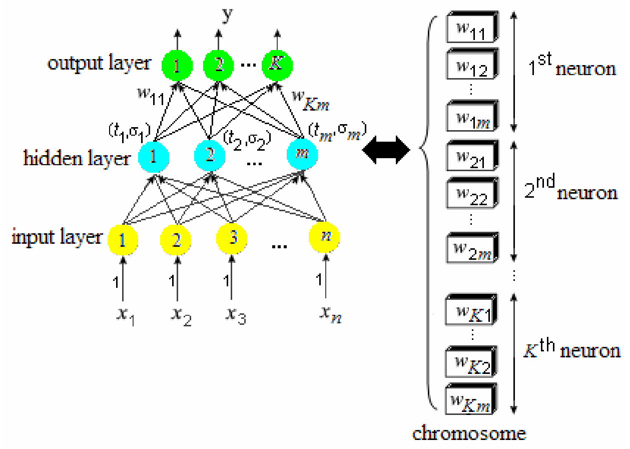


Fig.3: Genetic encoding technique used in case of neural weights fitting

The *fitness* used for each chromosome evaluation was calculated according to standard equation:

$$E = \left(\frac{k}{1+\text{MSE}}\right)\times\left(\frac{\text{max number of cycles}}{\text{current cycle}}\right),\ k = \text{ct}, \quad (7)$$

where MSE error was estimated for all $\{x_i\}_{i=\overline{1,P}}$ training patterns.

The ***stopping criterion*** was similar with the one used in case of the first genetic module. The ***selection*** of the parents for the next chromosomal population was also made using the well-known ***roulette*** algorithm.

The continuous ***crossover*** supposed the use of two random splitting points, and each chromosome had attached a certain crossover probability with values into [0.6, 0.95] range. By the same reasons as in case of first genetic module, the uniform ***mutation*** operator was also used.

To certainly obtain the best chromosomal solution, a technique quite similar with the above described ***pocket*** algorithm was also implemented.

More details related to the structure of the second genetic module used for optimization of the neural weights to output neural layer can be found in [5].

3 Experimental results

To add more consistency to the theoretical aspects treated in the first part of this paper, a real pattern recognition (PR) ***task*** was proposed to be solved (i.e., the PR task was to classify using video imagery, 5 input classes representing CCD images of some well-known military aircrafts).

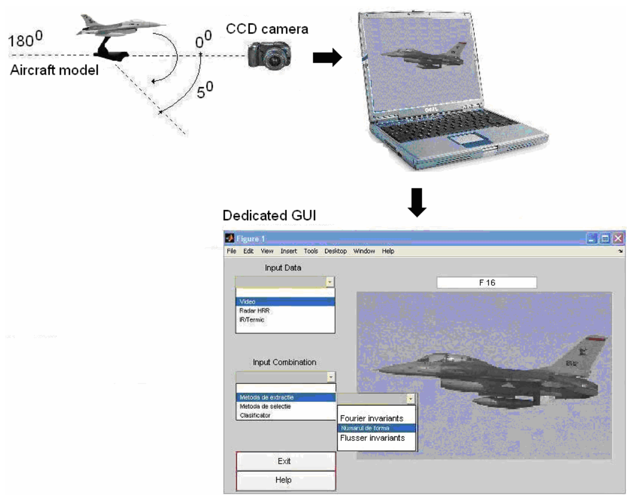
The video database was obtained using a (digital) photographical survey of *five* military aircraft models (i.e., F117, Mirage 2000, Mig 29, F16, and Tornado) scaled each at 1:48 (see Fig.4a). As one can see on Fig.4b, the survey was taken using a 5^0 increment in the azimuthal plane, using an angular range of $\left[0^0, 180^0\right]$, justified by the geometric aircraft shape symmetry. Consequently, a number of 37 video images/class is obtained. Also, each image from the input video database had a digital resolution of 520×160 pixels in an uncompressed *bmp* format.

Ÿ F117 model

Ÿ Mig 29 model

Ÿ F16 model

a) Examples of aircraft models used in video database design



b) Experimental setup

Fig.4: Database design

The acquisition and preprocessing algorithms used in case of video imagery were implemented in the form of a dedicated (MATLAB) GUI (see Fig.5) and were similar with ones described in [17].

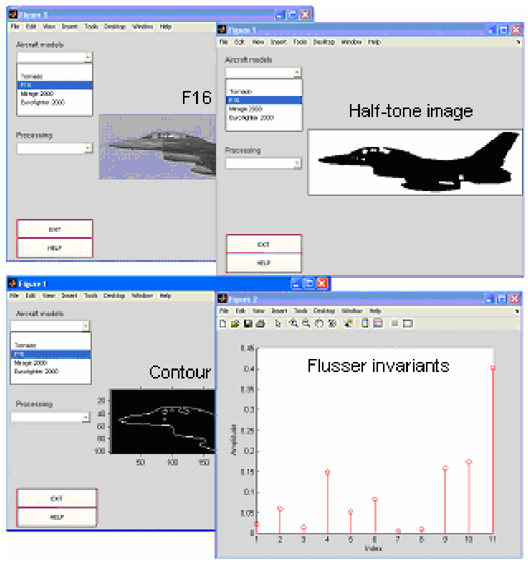


Fig.5: Dedicated GUI used to implement the acquisition and preprocessing stages

To implement the feature extraction stage, the *modified* Flusser invariants proposed in [18] were calculated.

Having as starting point the basic definition of the complex moments, and the content of Flusser theorem representing the background of Flusser moment family design [18], an improved version of standard Flusser moments, $\left\{\zeta_u\right\}_{u\in \mathrm{N}}$ can be obtained using the equation:

$$\zeta_{u\in \mathrm{N}} = \prod_{i=1}^{v} c_{p_i q_i}^{k_i}, \ \sum_{j=1}^{v} k_i \cdot (p_i - q_i) = 0, \qquad (8)$$
$$k_i, p_i, q_i \in \mathrm{Z}^+, v \geq 1$$

where v is the chosen number of moments (in our case, $v = 11$), and c_{pq} represents the $(p+q)^{\text{th}}$-order centered complex moment given by equation:

$$c_{pq} = \sum_{k=0}^{p}\sum_{s=0}^{q}\binom{p}{k}\binom{q}{s}(-1)^{q-s} \times j^{p+q-k-s} \ldots \qquad (9)$$
$$\ldots \times \xi_{k+s,\,p+q-k-s}, \quad p,q \in \mathrm{Z}^+$$

and ξ_{pq} represents the $(p+q)^{\text{th}}$-order shifted geometric moment defined according to [3].

Using equation (8), the new modified Flusser invariants can be rewritten as it follows:

$$\begin{cases} \zeta_1 = c_{11} \\ \zeta_2 = c_{21}c_{12} \\ \zeta_3 = \mathrm{Re}\left(c_{20}c_{12}^2\right) \\ \zeta_4 = \mathrm{Im}\left(c_{20}c_{12}^2\right) \\ \zeta_5 = \mathrm{Re}\left(c_{30}c_{12}^3\right) \\ \zeta_6 = \mathrm{Im}\left(c_{30}c_{12}^3\right) \\ \zeta_7 = c_{22} \\ \zeta_8 = \mathrm{Re}\left(c_{31}c_{12}^2\right) \\ \zeta_9 = \mathrm{Im}\left(c_{31}c_{12}^2\right) \\ \zeta_{10} = \mathrm{Re}\left(c_{40}c_{12}^4\right) \\ \zeta_{11} = \mathrm{Im}\left(c_{40}c_{12}^4\right) \end{cases} . \qquad (10)$$

The invariants set given by equation (10) has *four* useful properties: invariance at elementary geometric transformations (i.e., rotation, translation and scaling); discarding of the deficiencies assigned to standard central moments in case of symmetrical input images; it forms an independent base for pattern representation (i.e., a next feature selection stage is not necessary) and finally, increased robustness to the action of noisy factors (e.g., EW jamming etc.) inside of PR systems (e.g., ATR, ATTR systems etc.) comparing to the case of standard Flusser moments use. Consequently, using this improved version of Flusser moments, a serious increasing of the accuracy in pattern description is expected to be achieved.

More details regarding the design and properties of this invariants set can be found in [18].

Finally, after feature extraction stage, the vectors matrix for the next classification (recognition) purposes had 11×185 (i.e., 37×5) dimension.

The *main* objectives of this experimental paragraph were:

o1) to demonstrate the superiority at the performance level, of the proposed genetic optimization procedure comparing to others standard approaches used to train a RBF neural network;

o2) to compare at the performance level, the classification chain incorporating proposed optimized RBF network with the chain made by use of SART neural network.

To quantify and compare the performance level of different tested PR chains, as most important indicator, the *classification rate* (CR) has been computed. It is known that, the CR represents, in [%], the ratio between the number of correct classified patterns and the total number of these.

Related to the first experimental objective (see Fig.6), the full-genetic procedure used to optimize both connectivity and neural weights of RBF networks was compared as performance level (i.e., CR, training time etc.), to other standard (i.e., a supervised procedure for RBF center mapping and OLS algorithm, respectively) and improved (i.e., UCC algorithm proposed by Brown, and indicated in [3]) training techniques.

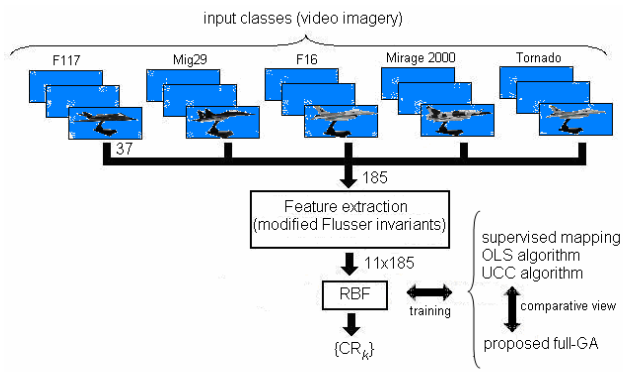


Fig.6: Block-diagram of first experimental objective

Using the available video image database, the CRs and other important parameters obtained after comparative study of these training methods, are synthetically presented in Table 1. Supplementary, in case of the proposed genetic technique, a minimal 2D projection of RBF network center mapping over input data space is shown in Fig.7.

Table 1

Center selection	**Classification performances** (CR and others training parameters)	
supervised mapping	90% m=8; K=5; 0.1 s; nepochs=10^4; $\varepsilon=10^{-4}$; σ=1	
OLS algorithm	92% m=10; K=5; 0.15 s; nepochs=10^4; $\varepsilon=10^{-4}$; σ=1	
UCC algorithm	93.5% m=11; K=5; 0.21 s; nepochs=10^4; $\varepsilon=10^{-4}$; σ=0.8	
proposed GA	95.5% m=14; K=5; 0.67 s nepochs=10^4 $\varepsilon=10^{-4}$; σ=0.8	GA modules
		155 s maxpop=50; maxgen=100 p_c=0.8; $\varepsilon_0=10^{-2}$
		121 s maxpop=50; maxgen=75 p_c=0.85; $\varepsilon_0=10^{-3}$; k=0.75

Experimental results

Having as starting point the experimental results reported in Table 1, a first preliminary *conclusion* is that one, the proposed genetic optimization procedure leads to an increased (on average) CR generally 3.6% more than other RBF network centers selection approaches. As can be seen in Fig.7, the first genetic module also leads to a very good center mapping over input data space (i.e., each significant data cluster had allocated at least a RBF center) even through it was used only a minimal 2D projection.

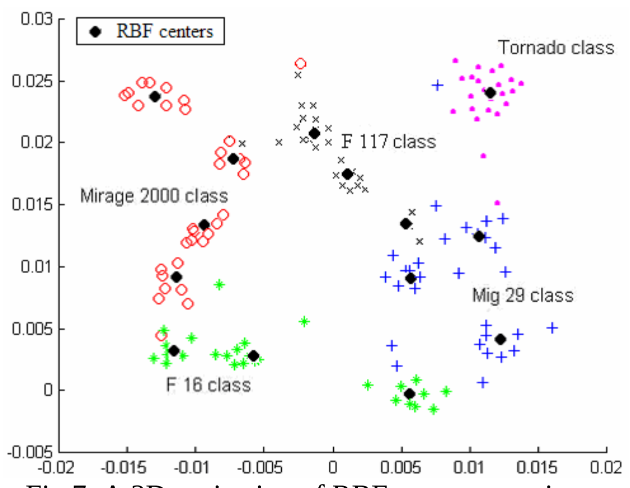


Fig.7: A 2D projection of RBF centers over input data space achieved by first genetic module

Finally, related to the second experimental objective (see Fig.8), the full-genetic optimized RBF neural network was compared as performance level, with SART (supervised ART) neural classifier. It is known that, an important property of SART network is that it needs no initial system parameter specifications and no prespecified number of center vectors, [19]. Because the number and the final values of the prototypes are automatically found during the training process for the SART network, this neural classifier represents in fact, an *improved* version of RBF (or LVQ) neural networks trained by standard approaches. More by token, the second experimental objective is one as soon as justified.

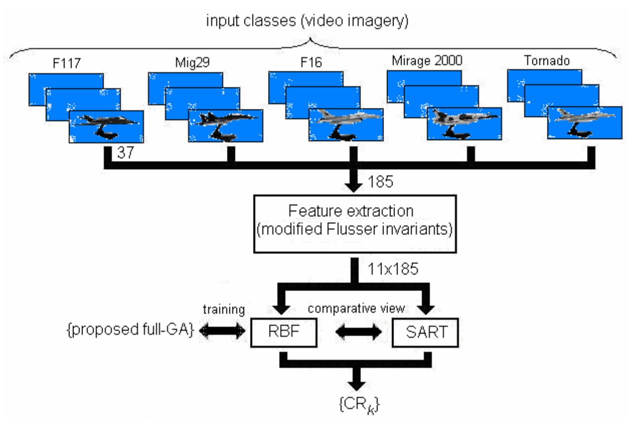


Fig.8: Block-diagram of second experimental objective

According to [19], SART neural classifier is developed using q* standard algorithm, [20]. It is used a set of prototypes which approximates the probability density modes of the vectors of each input class. The *NN* (*N*earest *N*eighbor) standard classification rule is then used to classify new vectors compared to these prototypes.

Generally speaking, this classifier generates in a *supervised* manner a new prototype if the distance between the new vector and the prototype existing yet, exceeds a threshold (according to *follow the leader* principle from ART neural network theory).

Consequently, the standard algorithm of SART neural classifier is presented in Fig.9. Finally, the basic steps describing the working algorithm of SART neural classifier are also presented below.

s1) *initialization stage*

- select randomly a vector as prototype for each class from input space;
- initialize the vector list associated to each class prototype;

s2) *training stage*

- while there are changed prototypes and the error rate is higher than a preset value and for each vector belongs to an input class, compares this vector with the actual prototypes and classified;
- if the input vector is correct classified, add the vector to the list assigned to the winner prototype.
- if the vector is not correct classified, declare the vector as new prototype and update the initial vector list;
- update the prototypes assigned to each class. Recalculate the prototypes as average between the vectors that are associated with its own list;
- check if some current prototypes have changed;
- eliminate the prototypes for that associated vector list contains only a vector;
- exit if the number of maxim iterations or prototypes is reached;
- calculate the neuronal weights of the output layer using the pseudo-inverse method or Widrow-Hoff algorithm [21].

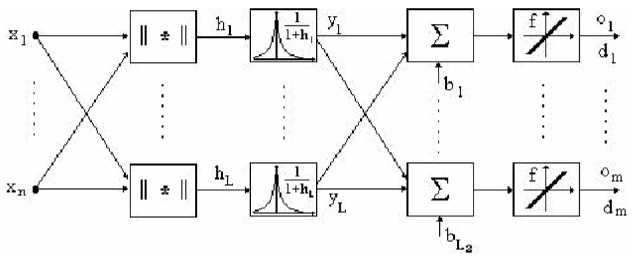


Fig.9: Standard architecture of SART classifier

More details regarding SART neural network can be found in [19] and [21].

Using the previous testing conditions, the CRs and other important parameters obtained after comparative study between the two classification chains made on one hand, by optimized RBF neural network and, on the other hand, by SART neural network are synthetically indicated in Table 2.

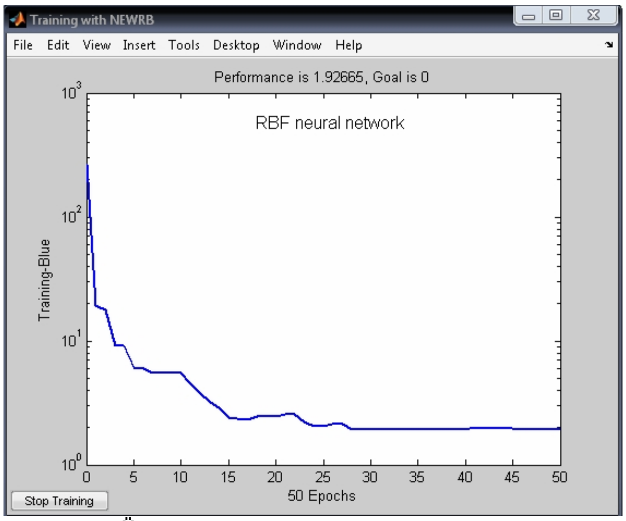
In addition to these experimental results, a graphical illustration of the most important aspects (training error, convergence property of training algorithms, classification results (CRs) etc.) related to training procedures used in case of two tested neural networks is shown in Fig.10.

Having as starting point the experimental results reported in Table 2, a second preliminary *conclusion* is that one, the CRs obtaining in case of two tested

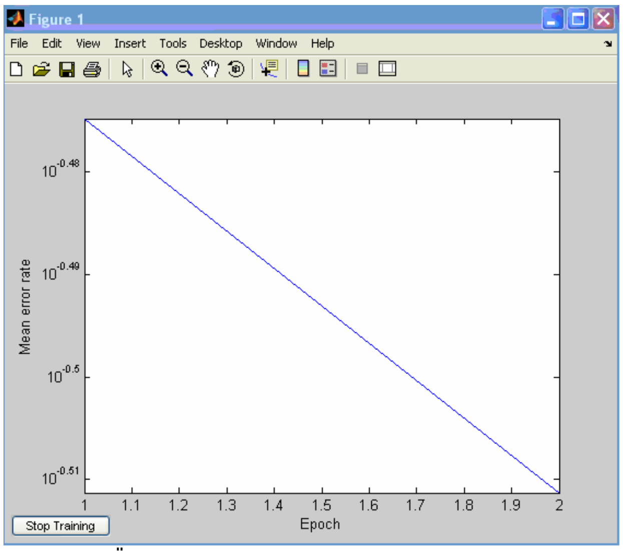
Table 2

Classifier	**Classification performances** (CR and others training parameters)
SART network	95%; 3.35 s maxepochs=25; maxprototypes=20; $\varepsilon=10^{-4}$
optimized RBF network	95.5%; 0.67 s m=14; K=5; nepochs=10^4; $\varepsilon=10^{-4}$; $\sigma=0.8$

Experimental results



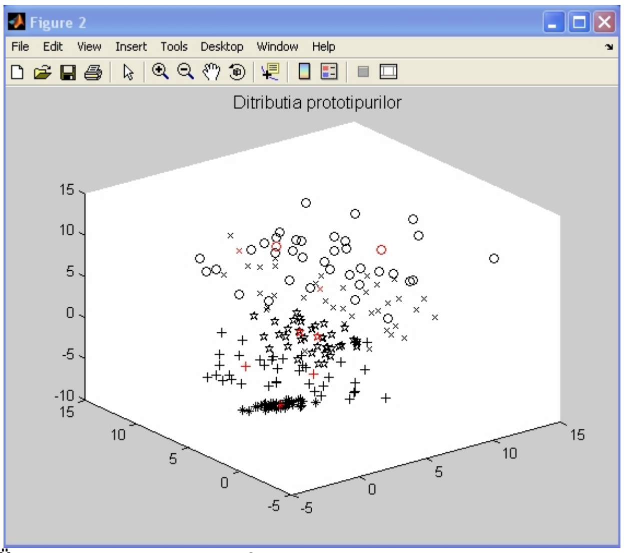
Ÿ Training error of RBF neural network



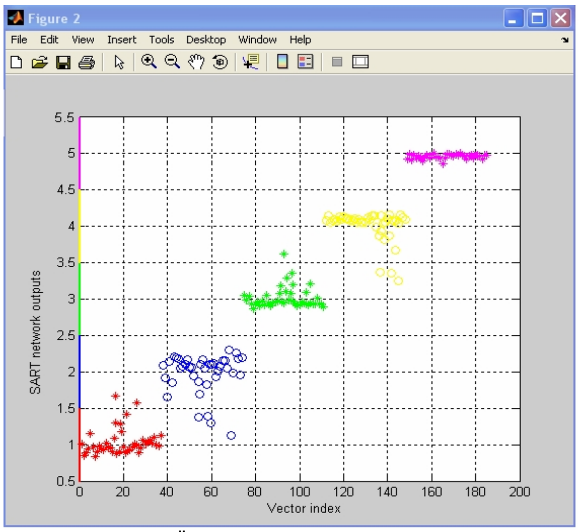
Ÿ Training error of SART neural network

neural networks are on average, similar. However, optimized RBF neural network turns out to be faster than SART network as training time and thus, more suitable to be (FPGA) hardware implemented inside of real-time classification chain (e.g., ATR, ATTR systems etc.), [22], [23].

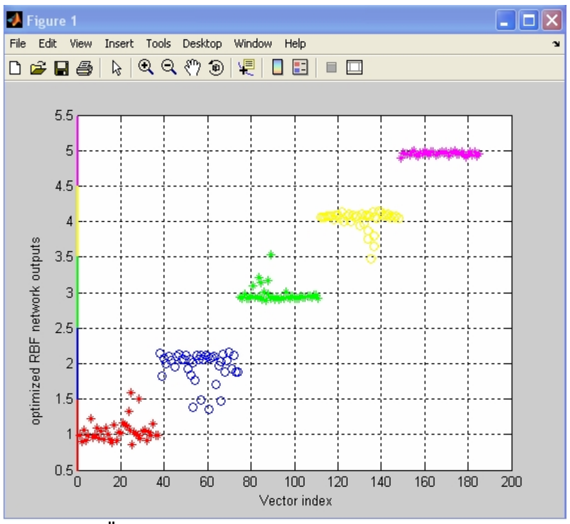
The applications described in this paragraph were implemented using specific functions from MATLAB toolboxes *nnet*, *image processing* and GAOT on a Pentium ^{TM} processor at 2.4 GHz.



Ÿ Prototype distribution of SART neural network (3D projection)
a) Training process results



Ÿ SART neural network



Ÿ GA-optimized RBF neural network
b) Classification results

Fig.10: Experimental results obtained in case of two tested neural networks

More details regarding experimental aspects treated in this section can be found in [5], [17], [18] and [19].

4 Conclusions and future work

The theoretical and experimental results presented in this paper lead to the following important *remarks*:
c1) the full-genetic procedure proposed to train RBF neural networks turns on to be as performance level, a correct option comparing to others standard (clustering) training techniques. In another train of thoughts, using a genetic searching of the RBF network parameters (i.e., connectivity, weights), the optimal property of this method is also assured;
c2) the optimized RBF neural network provides a classification level comparable with one assured by SART neural network, but this classifies much faster than the last one;
c3) finally, inside of (real-time) PR systems, the genetic optimized RBF network assures very good classification results (i.e., CRs more than 95%) and fully justifies as future potential option, its next hardware implementation.

In a future extension of this paper, the following important theoretical and experimental aspects will represent very interesting research directions:
fw1) the study (as performance level) of the possibility to implement the proposed classification chain (i.e., modified Flusser invariants and GA-optimized RBF network) inside of *multispectral* PR systems (e.g., using HRR, thermal or SAR imagery);
fw2) in order to increase the speed of genetic optimization process, another interesting point for a future development refers to a suitable *hardware implementation* of this genetic approach (e.g., based on FPGA devices use). Also, using this idea, the possibility to obtain the real values of proposed GA influence on the PR system performances (i.e., CR etc.) will become more relevant;
fw3) it is important to quantize the robustness degree of the proposed neural classification chain at the action of some common noisy factors (e.g., EW jamming, variable digital resolution etc.);
fw4) and finally, it is very interesting to extend the comparison (also as performance level) between full-GA optimized approach and other powerful techniques used for RBF network training.

References:

[1] C.M.Bishop, *Neural networks for pattern recognition*, Oxford University Press, 2004.
[2] L.V.Fausett, *Fundamentals of neural networks: architectures, algorithms and applications*, Prentice Hall, 1994.
[3] I.C.Vizitiu, *Neural networks used in video pattern recognition*, MTA Publishing House, 2002.
[4] M.S.Hassoun, *Fundamentals of artificial neural networks*, Massachusetts Institute of Technology, 1995.
[5] I.C.Vizitiu, *Neural networks and genetic algorithms. Theory and applications*, MTA Publishing House, 2004.
[6] E.G.Lacerda, *Model selection of RBF networks via genetic algorithms*, Ph.D.Thesis, Pernambuco Federal University, 2003.
[7] A.Saha, J.D.Keller, “Algorithms for better representation and faster learning in RBF networks”, *Advances in Neural Information Processing Systems,* vol. 2, 1990, pp. 482–489.
[8] I.C.Vizitiu, D.Munteanu, “A genetic procedure for RBF neural network center selection”, *Proc. of the 9th WSEAS International Conference NN’08*, 2008, pp. 37-40.
[9] D.Goldberg, *Genetic Algorithms in Serch, Optimization and Machine Learning*, Addison-Wesley, 1989.
[10] D.Dumitrescu, K.Simion, “Reducing complexity of RBF neural networks by dynamic evolutionary clustering techniques”, *Proc. of 11th Conference on Applied and Industrial Mathematics*, 2003, pp. 83-89.
[11] R.H.Sheikh, “Genetic algorithm based clustering: a survey”, *Proc. of 1st International Conference on Emerging Trends in Engineering and Technology*, 2008, pp. 314-319.
[12] D.K.Roy, L.K.Sharma, “Genetic k-means clustering algorithm for mixed numeric and categorical data sets”, *Journal of Artificial Intelligence&Applications*, vol.1, 2010, pp. 23-28.
[13] Y.Lu, S.Lu, “FGKA: A Fast Genetic k-means Clustering Algorithm”, *ACM Computer Survey*, 2004, pp. 26-32.
[14] L.O.Hall, I.B.Ozyurt, “Clustering with a genetically optimized approach”, *IEEE Trans. on Evolutionary Computation*, 1999.
[15] F.Wu, A.J.Kusalik, “Genetic weighted k-means for large-scale clustering problems”, University of Saskatchewan, Canada.
[16] R.M.Cole, *Clustering with genetic algorithms*, Master Thesis, University of West Australia, 1998.
[17] I.C.Vizitiu, F.Şerban, C.Molder, M.Stanciu, “Decision fusion method to improve the performances of multispectral ATR systems”, *Proc. of 1st WSEAS International Conference SENSIG’08*, 2008, pp.40-45.

[18] I.C.Vizitiu, D.Munteanu, "A new version of the Flusser invariant set for pattern feature extraction", *Proc. of 6th WSEAS International Conference CIMMACS'07*, 2007, pp. 244-249.
[19] I.C.Vizitiu, F.Serban, C.Molder, M.Stanciu, "An application of fuzzy-evolutive integral to improve the performances of multispectral ATR systems", *WSEAS Tran. on Information Science and Applications*, vol.6, 2009, pp.1893-1902.
[20] B.D.Ripley, *Pattern Recognition and Neural Networks,* Cambridge University Press, 2007.
[21] C.M. Bishop, *Pattern Recognition and Machine Learning,* Information Science and Statistics Series, Springer, 2006.
[22] A.Omondi, *FPGA implementation of neural networks*, Springer, 2006.
[23] I.C.Vizitiu, I.C.Rincu, I.Nicolaescu, "Optimal FPGA implementation of GARBF systems", *Proc. of 12nd Conference OPTIM*, 2010, pp.774-77.

Acknowledgments. The paper authors acknowledge the financial support of the CNMP Research Agency through the Project 11-029/2007.