# *Fingerprints Recognition Using a Neural Network*

*Roman E. López Bonilla, Ph.D.* [1]
*Department of Electrical Engineering*
*Polytechnic University of Puerto Rico*
*E-mail: rolopez@pupr.edu*

*Pedro Mayorga Ortiz*
*ITM de B. C. Mexico*
*E-mail: pmayorga@bestit.itmx.mx*

## *ABSTRACT*

*A fingerprint recognition approach, based on a backpropagation algorithm using a neural network, is presented. The purpose of our work is to show the relevance of the fingerprint method characterization by using the vectorization method. This method reduces dramatically the required amount of data for later application on neural networks having some variations of the backpropagation method for fingerprint recognition. When this step is completed the next step is to establish a comparison between the algorithms in terms of time. The algorithms were implemented by using 'C' language on an IBM PC's platform. The neural network has four layers. Two of them are hidden. The third one is the input and the last one is the output. In our particular experiment, we used five neurons in the input layer and seven neurons in the output layer. This neural network could be trained to a maximum capacity of 128 fingerprints. However, the method could be easily expanded for the recognition of a larger amount of fingerprints. The performances of the algorithms were checked against the run time for each algorithm using vectors extracted from digitized fingerprint images. The LMS algorithms have several variants and each one of them has their advantages. In this case, the reference for comparison is the computation time, done with the same computer, and for the same application.*

## *SINOPSIS*

*En este artículo se presenta un método para el reconocimiento de huellas dactilares basado en una red neuronal usando técnicas de retropropagación. El propósito de este artículo es mostrar las características relevantes de la caracterización de una huella dactilar por el método de vectorización. Este método reduce drásticamente la cantidad de datos de la huella dactilar. Los datos obtenidos por el método de sectorización son alimentados a una red neuronal usando algunas variantes del método de retropropagación.*

## *I- INTRODUCTION*

Recent research shows that, because of consistencies, an architecture of high order can be built from neural networks and can be effectively used on pattern recognition problems, which on one plane are invariant to rotation, scale and translation, all of this in a gray scale level [4, 5, 7]. A one single layer network is bounded to linear problems. But in multilayer networks, the use of backpropagation is needed to solve multiple non-linear problems among which we found that pattern recognition is invariant to distortion [4, 7].

However, although backpropagation requires a large number of iterations or epochs, it may achieve 100% accuracy [5].

One way to reduce the quantity of points when working with fingerprint images is the minutiae extracted method. Here is where neural network architecture may be applied. The NN (neural network) is more useful when applied on problems that are easy to solve for us humans and difficult to solve for the computers. Problems of this kind can be extraction feature and pattern recognition, which is the case of fingerprints in gray scale level [5,7]. These characteristics are bifurcations or wrinkles present on the fingertip and they are unique for each human being.

On the present work we used the central points from the minutiae as the data in the vectorization process, which result in vectors of five elements each one. Then, the neural network was trained with these minutiae or vectors by teaching it to recognize a fingerprint using these vectors [7, 8]. The network was designed with seven outputs for a capacity to identify 128 fingerprints, and may be redesigned by adding more outputs in order to

[1] Main prof. CITEDI-IPN on sabatic

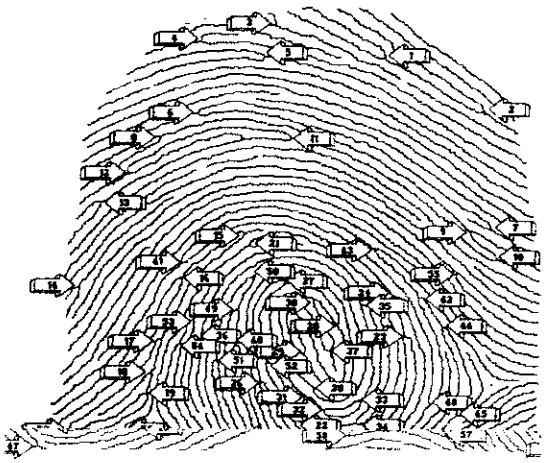

***Figure 1.*** *A fingerprint with arrows pointing to minutiae.*

increase its capacity considerably. For test purposes we used six different LMS algorithms.

## *II- FINGERPRINTS*

When we have a fingerprint image, we have also a very large matrix, where each point in the matrix is a data to work with. Fortunately, the fingerprints can be characterized by minutiae; therefore we need a smaller quantity of points to represent all the minutiae for a fingerprint image.

As we can see in Figure 1, there are only 58 points of minutia, while the quantity of points that we need to represent the whole image is bigger. There is another problem: when a fingerprint image is captured, it is rarely repeated because it is not probable for a person to put his/her finger on the same place and in the same position. Therefore, it should not be expected to obtain the same fingerprint image twice because of variants such as rotation and translation. However, by applying a vectorization process [7, 8] over the fingerprint image, as we see in the Figure 2, we can eliminate this problem.

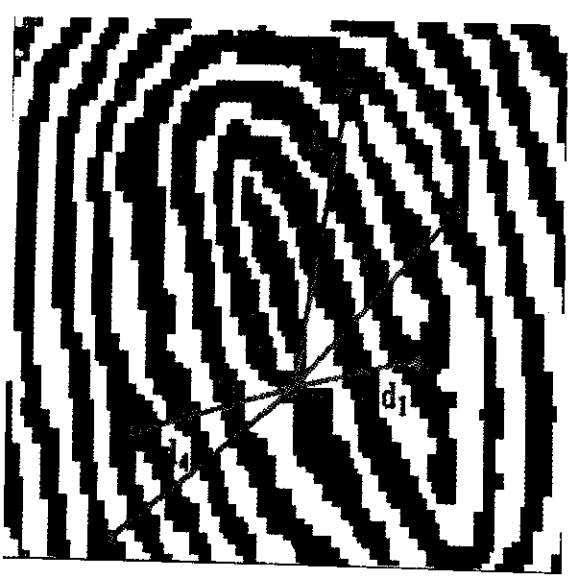


***Figure 2:*** *Vector element of one vector of minutiae, each vector has five elements d1, d2, d3, d4, d5*

As shown in the Figure 2, each vector takes a minutia point and the five closer minutia points in order to do a vector.

### *III- BASIC EQUATIONS*

The basic equations used for all the calculations made in our experiment are as follows:

$$E_p = \frac{1}{2}\sum_{j=1}^{n}\left(d_j - x_j\right)^2$$

$$E_p = \frac{1}{2}\sum_{j=1}^{n} e_j^2 \qquad (1)$$

Equation 1 means the sum of the square of the difference between network output ($x_j$) and the desired response ($d_j$), where j is the index of each of the elements from the output vector and the desired output vector and $E_p$ is the half sum of the square of the errors ($e_j$). Note that equation 1 applies only to the final layer. In order to obtain the local errors in the hidden layers, we need to compute the error factor $\delta_j^l$ for each hidden layer $l$. To compute the local error for the third hidden layer we use the following equation:

$$\delta_j^3 = e_j \frac{\delta X_j^3}{\partial \upsilon_j^3}$$

$$\delta_j^3 = \left(d_j - x_j^3\right)x_j^3\left(1 - x_j^3\right) \qquad (2)$$

To compute the local error for the second hidden layer we use the following equation:

$$\delta_j^2 = \frac{\partial x_j^2}{\partial \upsilon_j^2}\sum_{p=1}^{n}\delta_p^3 w_{j,p}^3 \qquad (3)$$

and to compute the local error for the first hidden layer we use the following equation:

$$\delta_j^1 = \frac{\partial x_j^1}{\partial \upsilon_j^1}\sum_{p=1}^{n}\delta_p^2 w_{j,p}^2 \qquad (4)$$

Hence, using the sigmoid function, we obtain:

$$\frac{\partial x_j^l}{\partial \upsilon_j^l} = x_j^l\left(1 - x_j^l\right) \qquad (5)$$

where $x_j^l$ is defined as:

$$x_j^l = \frac{1}{1 + \exp\left(\sum_{k=1}^{N_k} w_{j,k}^l x_k^{l-1}\right)} \qquad (6)$$

As we can see in the previous equations, the local error in the hidden layers is a function of the local errors of higher layers. The bigger difference in the learning rules for hidden layers and higher layers is the way of evaluating local errors (note that when we talk about higher layer we refer to the output layer). In the output layer the error is a

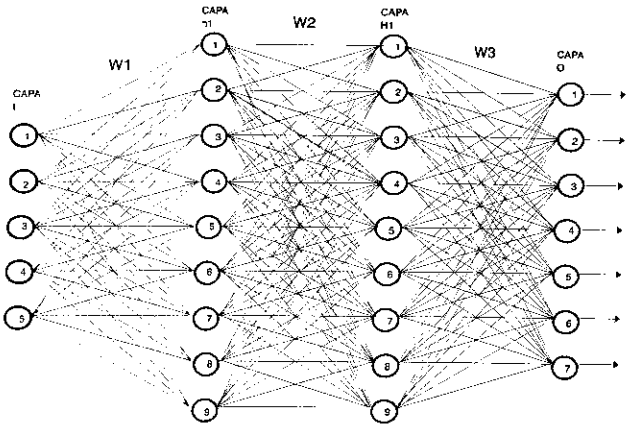


***Figure 3:*** *Architecture of ANN for the recognition of fingerprints with four layers: one input layer, two hidden layers and one output layer. The input layer does not make processing. The two hidden layers process the data and the last layer gives the code number of the fingerprint.*

function of current output, and the desired output multiplies the calculation of the derivation of the current output (equations 1 and 2).

In the output layer the error is a function of current output and the desired output multiply the calculation of derivation of current output, in other words, the calculation of the derivation of the activation function or calculation of the derivation of the sigmoid function in the neuronal which we are talking (equation 1 and 2). In the hidden layer the error is computed using the error factor calculated before in equations 3 and 4.

Before we can concentrate in our specific problem and talk about the different algorithms, let us see the architecture of the ANN (artificial neural network).

In Figure 3 we used the letter I for the input layer, h1 for the first hidden layer, H1 for the second hidden layer and O for the output layer. Next, we will be using the terms X1, X2, X3 and X4 for each of the layers respectively. In the other hand, we are going to be using the terms W1, W2 and W3 for the matrix of synapses weights between layers with X1, X2, X3, and X4 as the neural vectors for each layer. Figure 3 shows circles within numbers meaning the number of neural nodes in its vector. This number of neural nodes on the input layer is so because our fingerprint vectors has five elements. The seven neural nodes on the output layer represents the capacity of recognition of the system, which can recognize up to 128 different fingerprints. As we saw before, there are three Wj bidimensional weighted matrices. These matrices are W1 (a 5X9 matrix), W2 (a 9X9 matrix) and W3 (a 9X7 matrix).

## *IV- LMS ALGORITHMS*

So far we have been establishing the elements and the basic equations for the network [6,8], but we need to take a closer look into our project. Therefore we will add algorithms and flow lines to the network to represent the process flow. In particular we will use the backpropagation method and LMS algorithms for the adjustment of weighted synapses. It is important to note that this adjustment has to be made only during the learning process or, in other words, when the net is in the process of training to recognize fingerprints. It is important to note also that in this process of learning is where most of the time is consumed, but once the net has "learned" to recognize fingerprints, it is faster to recognize them later. If the net "knows" a fingerprint, it is because the weight matrix has been adjusted. There are several LMS algorithms that we are going to use. Let us first explain LMS standard:

1. **LMS Standard** is a useful algorithm used in the literature [5]. This algorithm is used in Backpropagation to adjust the weights in the network when it is trained. The Backpropagation comes from the Delta Rule that is a particular case of the error gradient and the Chain Rule from elemental calculus [8]. This method adjusts the weight as we saw earlier using the error in high layers and backward to lower layers in neural networks, as we can see in the Figure 4.

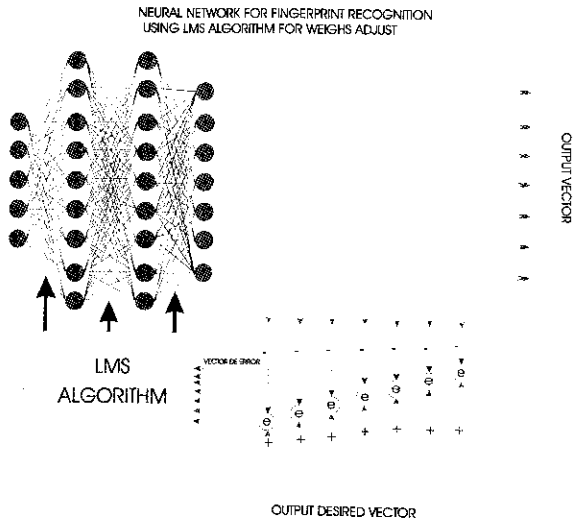


***Figure 4:*** *Neural Network with Backpropagation using LMS algorithm for fingerprint recognition*

The equation for the LMS standard is as follows:

$$W_{i,j}^{l} = W_{i,j}^{l} + \alpha X_i^{l-1} \delta_j^{l} \quad (7)$$

The i index is for the neuronal from the lower layer while the j index is for the neuronal in the higher layer. The l upper index means weight matrix. Let us assume that i=2, j=5, and l=1. In this case we are talking about the synapses element between neuronal 2 in the layer 1 and neuronal 5 in the layer 2 because l represents the weight matrix between layer 1 and 2. The factor $\alpha$ is a learning factor which typically is $0<\alpha<1$. In our case we choose $\alpha=0.5$ [6]. X is a neuronal vector where the indexes have the same meaning as before. I is the neuronal element from the neuronal vector X. The last term $\delta$ is called "Error Correction Factor" and is used to calculate the error in lower layers as stated previously. The meaning of the indexes and of the superscripts on the indexes for the remaining algorithms does not change.

2- **Normalized LMS:** This algorithm uses the square of the magnitude of the neural vector from the lower layer plus a factor $\beta$ for the adjustment of the second term that we see in the algorithm LMS standard. The boundary for $\beta$ and $\eta$ are as follows: $\beta \geq 0$, $0<\eta<2$. Typically $\eta$ is between $0.1 \leq \eta < 1$, and for $\beta$ we choose 0.5. The equation that represents this algorithm is as follows:

$$W_{i,j}^{l} = W_{i,j}^{l} + \frac{\eta X_i^{l-1} \delta_j^{l}}{\beta + \left\| X^{l-1} \right\|^2} \quad (8)$$

Here $\eta$ is a learning factor that modulates the amount of loops for the adjustment of the weight matrices so the net can "know" or learn. In other words, if $\eta$ is small, the amount of epochs is going to be large. But if $\eta$ is very large, the quantity of epochs is going to be too small for the net to learn to identify something.

3- **LMS Algorithm with Momentum** introduces changes with respect to the LMS standard; and this is exactly the momentum that we can see on the following equation:

$$W_{i,j}^{l} = W_{i,j}^{l} + \eta \delta_j^{l} X_i^{l-1} + \alpha [W_{i,j}^{l} - AW_{i,j}^{l}] \quad (9)$$

The third term in equation 9 is given for the learning rate ($\alpha$), which multiplies the difference between W and AW. The term AW is the value of W calculated one iteration before. In the first calculation, or iteration, AW is equal to zero. The algorithm is as follows:

Initialization:

$$Temp_{i,j} = W_{i,j}^{l} = rand(value); \quad (10.a)$$

$$AW_{i,j}^{l} = 0 \quad (10.b)$$

Adjusting equation:

$$W_{i,j}^{l} = W_{i,j}^{l} + \eta \delta_j^{l} X_i^{l-1} + \alpha \left[ W_{i,j}^{l} - AW_{i,j}^{l} \right] \quad (11.a)$$

$$AW_{i,j}^{l} = Temp_{i,j} \quad (11.b)$$

$$Temp_{i,j} = W_{i,j}^{l} \quad (11.c)$$

Here we use a temporal variable called *"Temp"*, which is used to take the preceding value of *W*.

4- **Smoothed LMS** is very similar to the LMS algorithm with momentum (equations 10 and 11) except for the factor $(1-\alpha)$, which also modulates the learning rate. Therefore the smoothed LMS algorithm equations are similar to those of the LMS algorithm with momentum. The equations are the following:

Initialization:

$$Temp_{i,j} = W_{i,j}^{l} = rand(value); \quad (12.a)$$

$$AW_{i,j}^{l} = 0 \quad (12.b)$$

Adjusting equation:

$$W_{i,j}^{l} = W_{i,j}^{l} + \eta(1-\alpha) \delta_j^{l} X_i^{l-1} + \alpha \left[ W_{i,j}^{l} - AW_{i,j}^{l} \right] \quad (13.a)$$

$$AW_{i,j}^{l} = Temp_{i,j} \quad (13.b)$$

$$Temp_{i,j} = W_{i,j}^{l} \quad (13.c)$$

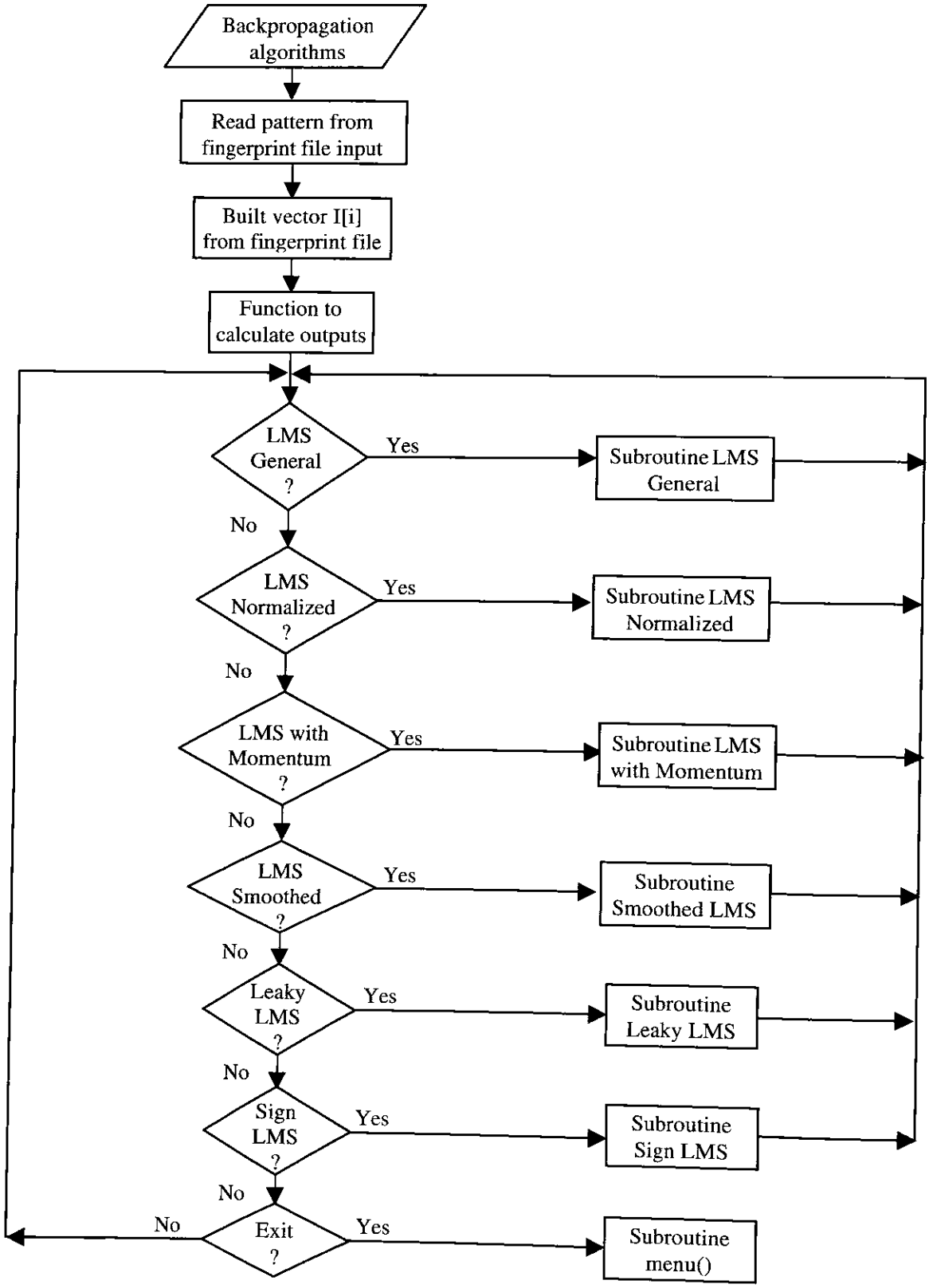


***Figure 5:*** *Main body of the Backpropagation program with LMS Algorithms*

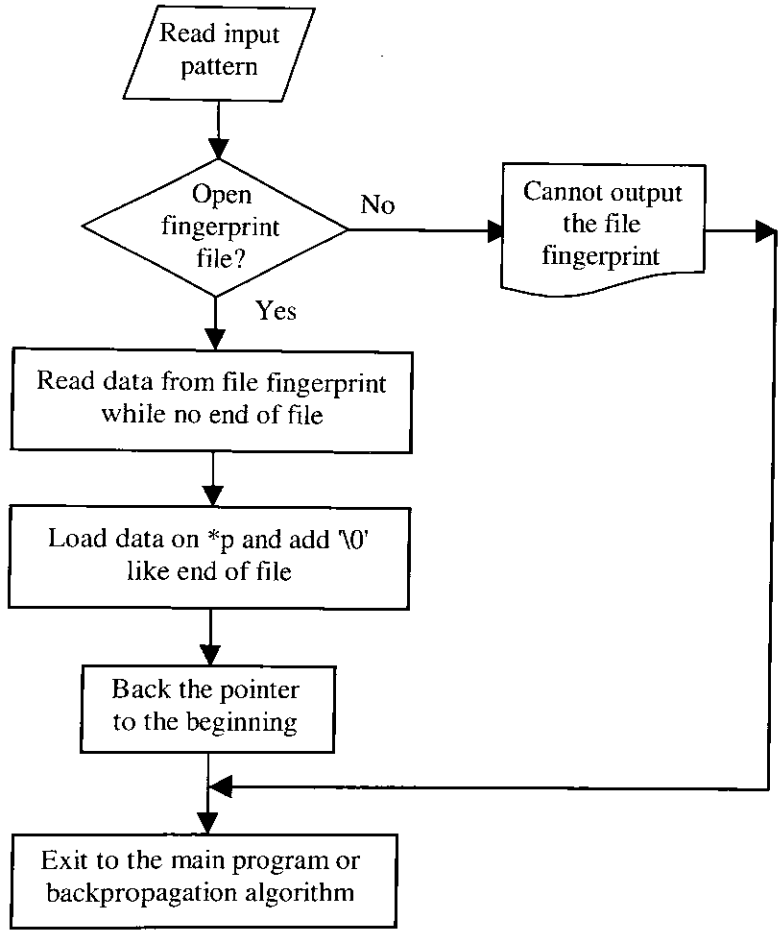


***Figure 6:*** *Function to read fingerprint data file*

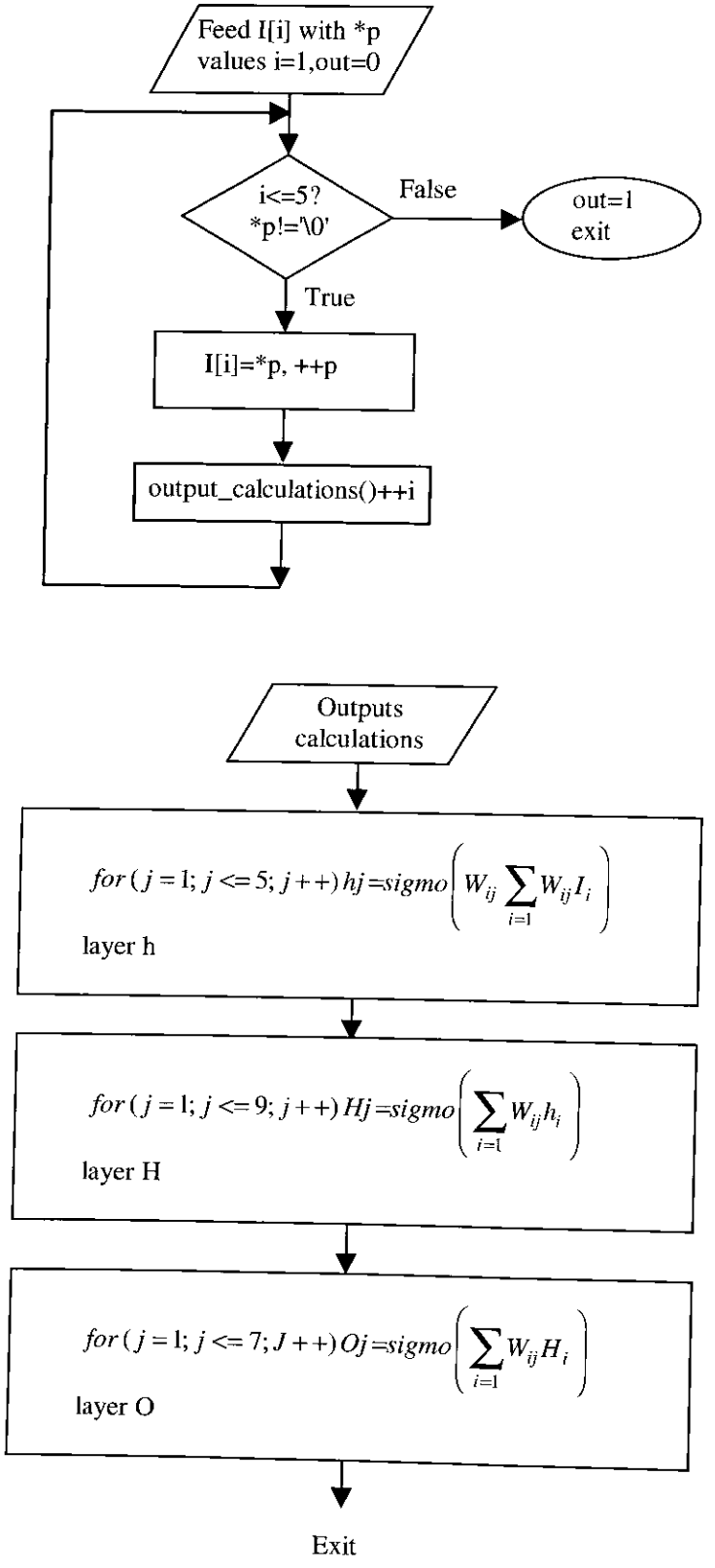


***Figure 7:*** *Functions to compute the outputs*

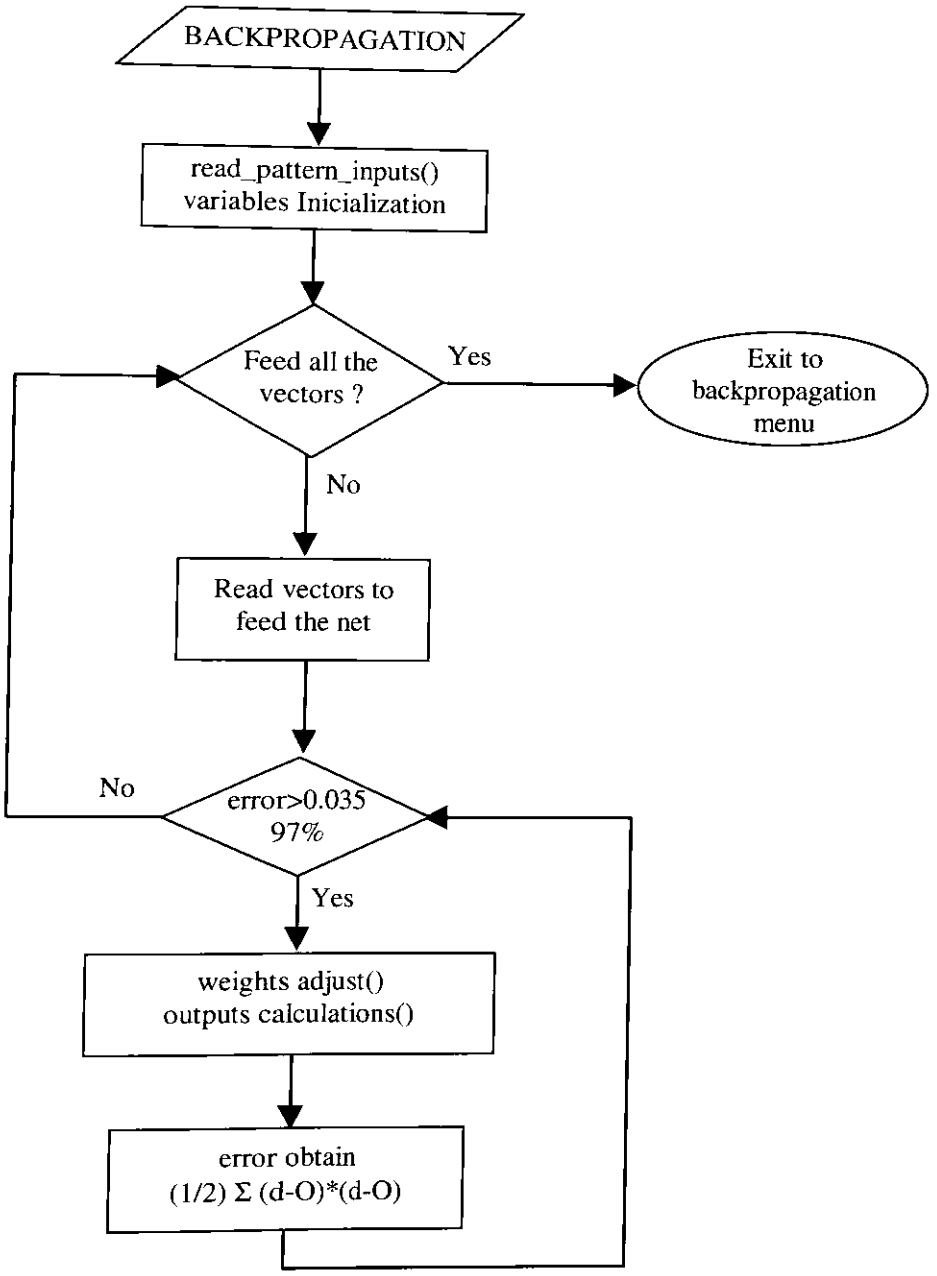


***Figure 8:*** *Backpropagation subroutine with standard LMS*

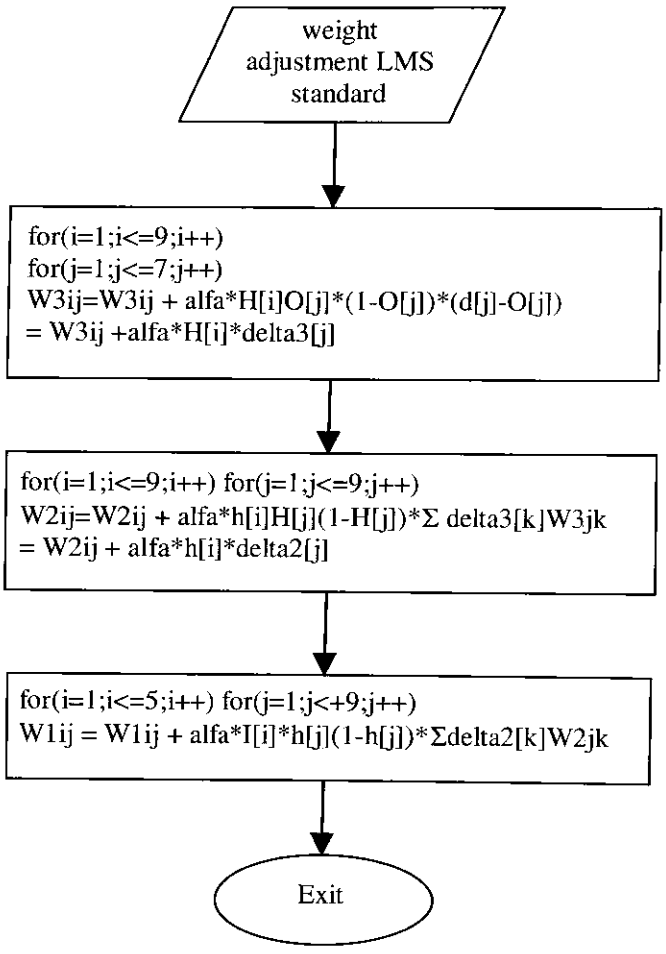


***Figure 9:**. Weight adjustment function using LMS standard*

5- **Leaky LMS** algorithm is as follow:

$$W_{i,j}^{l} = (1-\gamma)W_{i,j}^{l} + \eta\delta_j^{l} X_i^{l-1} \qquad (14)$$

where $\delta$ is calculated as before. There is a new term $(1-\gamma)$ that is also used as a learning rate term.

6- Finally the **Sign LMS** algorithm, which takes its name from the sign function that is used in the algorithm to adjust the weights. The equation is as follow:

$$W_{i,j}^{l} = W_{i,j}^{l} + Sign(\delta_j^{l}) X_i^{l-1} \qquad (15)$$

The sign function returns a plus or a minus sign when the expression is evaluated. If the value of the expression is less than zero, it must return a minus sign. Else, a plus sign must be returned. The equation is shown below:

$$Sign(\delta) = \begin{cases} +1\_if\_\delta \geq 0 \\ \\ -1\_if\_\delta < 0 \end{cases} \qquad (16)$$

## *V- RESULTS*

The algorithms presented here use about the same amount of terms to compute the synapses weights of the neural network. In particular, the LMS standard needs 423 epochs or iterations needed to compute or adjust the weighted matrices. Normalized LMS needs 348 epochs and the Sign LMS uses 348 epochs. With the remaining three algorithms we had overflows and some other problems. For all the algorithms we used the same degree of accuracy of 96.5% and the same computer, which is a 133 MHz Pentium PC platform. The learning process was developed in 'C' language.

## *VI- DISCUSSIONS AND CONCLUSIONS*

Although we still have to do some work on some algorithms, we can modify and test them with different degrees of accuracy in order to obtain better results. However, the most important thing in our experiment is that we demonstrated that it is possible to build a reliable system that uses few points and fewer data for fingerprint recognition. The quantity of synapses weights that we used to identify 128 fingerprints was 189. As we can see in Figures 3 and 4, if we increase a few outputs, the capacity of the network will be increased substantially. However, the flow diagrams (see Figures 5, 6, 7, 8 and 9) do not have to be modified significantly. Regarding runtime, when recognizing fingerprints, the learning process takes more time than the recognition process. Nevertheless, the learning process is performed only once.

## *VII- REFERENCES*


1- Simon Haykin "*Neural Networks Expand SP's Horizons*", IEEE Signal Processing Magazine, March 1996, pp. 1053-1058

2- Brigitte Colnet and Paul Bertrand, "*A Neural Network Approach Processing*"

3- Mario Mastriani, "*Self-restorable Stochastic Neurocontrol using Back-propagation and Forward-propagation Training Algorithms*", Secretaría de Investigación y Doctorado (SECID), Facultad de Ingeniería de la Universidad de Buenos Aires (FIUBA)

4- Lilly Spirkovska and Max B. Reid, "*Connectivity for Higher-order Neural Networks Applied to Pattern Recognition*", NASA Ames Research Center, Intelligent Systems Technology Branch M/S 244-4 Moffett Field, CA 94035-1000

5- Ming-Tak Leung, W. E. Engeler and P. Frank, "*Fingerprint Processing using Backpropagation Neural Networks*", Electronic Systems Laboratory, CRD, Schenectady, NY.

6- Cichocki, R. Unbehauen, "*Neural Networks for Optimization and Signal Processing*", B. G. Teubner Stuttgart, John Wiley & Sons, 1993.

7- López Bonilla Román E., Mayorga Ortiz Pedro "*Reconocimiento de huellas digitales usando la técnica de retropropagación en una red neuronal*", VII Congreso Inter-Universitario CIECE'97.

8- Rafael C. González, Richard E. Woods, "*Digital Image Processing*", Addison Wesley, September, 1993

9- M. Election, "*Automatic Fingerprint Identification*", IEEE Spectrum, vol. 10, pp. 36-45, September 1973

10- Bijan Moayer and King-Sun Fu, "*A Tree System Approach for Fingerprints Pattern Recognition*", IEEE Transactions of Pattern analysis and Machine Intelligence, Vol. PAMI-2, No. 3, May 1980, pages 223-231.

11- Trujillo S., "*El estudio cientifico de la dactiloscopía*" De Limusa, México, 1992.