# LEFT VENTRICLE DETECTION IN RADIONUCLIDE VENTRICULOGRAPHY BY A MODEL OF NEURAL NETWORK

J. DAMIEN, M.S., L. ITTI, P. EGROIZARD, M.D., R. ITTI, M.D., Ph.D.
Centre de Médecine Nucléaire, BP Lyon-Monchat, 69394 Lyon Cédex 03

***Abstract*** **- A 2-layer neural network was applied to determine the LV in radionuclide ventriculography. After learning by back-propagation, the correlations between computed pictures and learning data-set outputs, and between learning data-set outputs and other pictures were excellent (r=0.92 and r=0.83 respectively).**

## I - INTRODUCTION

Nuclear medicine has always been precursory in the image processing field because of the intrinsic features of the pictures. Hence, we planned to apply a 2-layer perceptron to detect the left ventricle (LV) in radionuclide ventriculography. The goal of this paper is to present the results of our device, which final aim is to determine automatically the Left Ventricular Ejection Fraction (LVEF) and the time-activity curve being clinical parameters of cardiac function.

## II - NEURAL NETWORK STRUCTURE

The neural network consists of two layers designed by an intermediate and an output layer. Each layer has 1024 neurons and each neuron has 1024 weights and a bias (which is not an output of the previous layer) as inputs. The transfer function of each neuron is a sigmoid. The output of each neuron is the sum of the weighed inputs added to the bias, modulated by the transfer function. Each neuron is connected with all neurons of the previous layer.

## III - LEARNING OF THE WEIGHTS OF NEURAL NETWORK BY THE BACK-PROPAGATION METHOD

### *1 - Learning Data-Set*

We studied 15 series of 16 pictures, describing the total cardiac cycle, from which the diastolic picture was extracted. The original idea was to present 64 x 64 pixels rough radionuclide pictures as learning data-set inputs (each pixel corresponding to an input). However, because of the architecture of our network, the number of connections was far too big (2 x 4096 x 4096) for the power of the used computer (IBM RS 6000 - 320). Hence we reduced the network height by keeping a 32 x 32 pixels area focused on the LV. The pictures had been previously smoothed with a temporal filter (arithmetic average of pixels from 3 successive pictures), and with a spatial filter (classic 9 points filter). The output pictures were obtained after an expert had defined the LV area. The pixels belonging to this region kept their activity level, all others were put to 0. So, our learning data-set included 15 input and 15 output pictures.

**Table 1**
**Notations used in section I - 2**

| | input i | intermediate j | output k |
|---|---|---|---|
| learning data | $\mathfrak{E}$ | | $\mathfrak{S}$ |
| weight | | $w_{ij}$ | $W_{jk}$ |
| bias | | $b_j$ | $B_k$ |
| computation | | $t_j=\Sigma_i(w_{ij}\,\mathfrak{E}_i+b_j)$ | $Tk=\Sigma_j(W_{jk}s_i+b_j)$ |
| | | $s_j = f(t_j)$ | $S_k = f(T_k)$ |
| errors | | $e_j$ | $E_k$ |

### *2 - The Back-Propagation Method*

Among the numerous learning algorithms that have been proposed for complex connectionist networks, Back-Propagation (BP) is probably the most widespread. BP was proposed by Rumelhart et al in 1986[1], but this method had been developed earlier by several independant groups [2-4]. In this section, the method is briefly described, but the equations are adapted to our problem (all notations are in table 1).

For each output neuron, the error is computed from the difference between the value calculated during the forward propagation, and the wished value, which is a learning data-set output. The correction to make has of course to be proportional to the difference already cited, but also to the derivative of the transfer function at this place (1), to take into account the slope of this transfer function.

$$E_k = (\mathfrak{S}_k - S_k)\, f'(T_k) \qquad (1)$$

The weight of the considered neuron is adjusted by substracting a value $\Delta w_{ij}$ proportionnal to the error from the current weight (2). By doing this, the higher the output of the *j*th neuron is, the more important the correction is.

$$\Delta W_{jk} = E_k\, s_j \qquad (2)$$

For the weights of the intermediate layer, we wanted the correction to attenuate the neurons most likely to induce the output errors (3).

$$e_j = f'(t_j)\, \Sigma_k\, W_{jk}\, E_k \qquad (3)$$

The local error is proportional to the derivative of the transfer function and to the weighed sum of the output errors.

Then we adjust the weight with (4)

$$\Delta w_{ij} = e_j x_i \qquad (4)$$

The output error should decrease gradually after each successive iteration, because of the adjustment of the weights and the bias. The arrest condition should be an error level, considered as acceptable, or a predefined number of iterations (in our case 2010). The end test was performed after a global data-set run. The picture presentation order being a very important parameter for the speed of the algorithm convergence, a pseudo-aleatory computing was carried out to design the order.

## IV - RESULTS

### *1 : Algorithm Convergence*

Figure 1 materialises the evolution of the mean error. This parameter is the sum of the half square errors (difference between computed and wished values) which are computed for each output layer neuron. Subsequently, it is divided by the number of iterations. The global feature of the curve shows that the decrease is very fast until the 50th iteration. The decrease is much slower, afterwards.

### *2 : Network Efficiency*

After each backward phase, we added a forward propagation to simulate the normal use of the neural network. The computed errors for 3 different pictures, belonging to the learning data-set, are shown on Figure 2.

The curves correspond to the biggest LV, mean LV and smallest LV, respectively. The most important characteristic is the large oscillations for the extrem LV sizes.

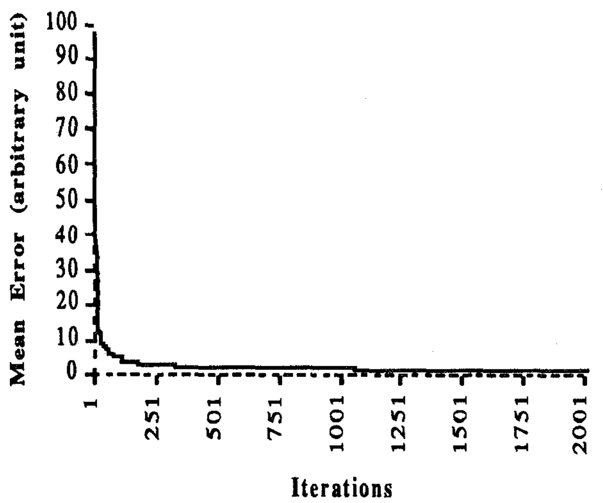

**Figure 1 : Mean Error Evolution During Learning**

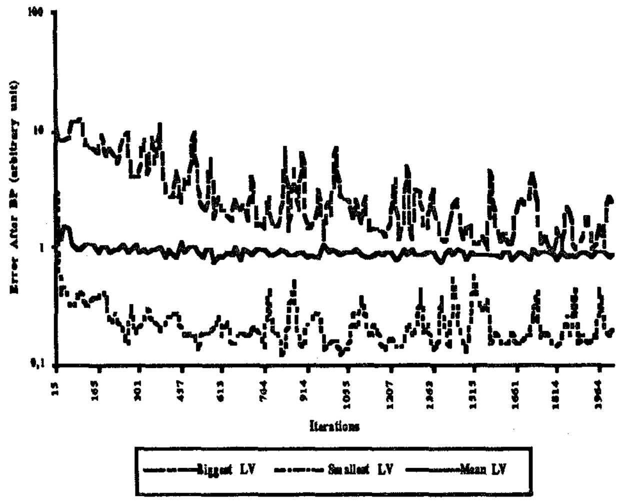

**Figure 2 : Evolution of the efficiency of the neural network**

Based on the data of figure 1, one could presume 100 iterations to be enough. But the pursuit of the learning is useful for the LVs which are far from the mean. At last, this figure gives us an other condition for ending the learning. Indeed, it could be very dammageable if the learning stops when the errors of the biggest and smallest LVs are maximal.

We must add an other test to evaluate correctly the efficiency of our neural network, for there is no quality control on the LV detection from a picture which does not belong to our learning data-set. The correlation was computed from the number of pixels belonging to the LV and keeping their activity level +/- 10 %. The results show r=0.92 with learning data-set pictures and falls to r=0.83 with these latter added to 30 new pictures.

## V - CONCLUSION

Our neural network seems to work properly. However, the efficiency is not proved, if the center of the LV does not correspond to the center of the picture. A Kohonen network should give better results.

REFERENCES

[1] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. Nature 1986 ; 323, 533-536
[2] A. Bryson and Y. HO. Applied optimal control. 1969. Blaisdell Publishing Co.
[3] Y. Le Cun. A learning scheme for asymmetric threshold network. In Bienenstock, E., Fogelman-Soulié, F., and Weisbuch, G., editors, Disordered systems and biological organization, 233-240, Les Houches, France, Springer-Verlag
[4] P. BALDI. Back-propagation and unsupervised learning in linear networks. In : Back-propagation : theory , architectures and applications. Y. Chauvin and D.E. Rumelhart (Eds). L. Erlbaum. In press