A USER-FRIENDLY AUTOMATIC TOOL FOR IMAGE CLASSIFICATION BASED ON NEURAL NETWORKS

B. Buttarazzi, F. Del Frate* , C. Solimini

Università Tor Vergata, Ingegneria DISP, Via del Politecnico 1, I-00133 Rome, Italy
delfrate@disp.uniroma2.it

KEY WORDS: Neural networks, automatic classification, image processing

ABSTRACT:

In this paper the implementation and the functionality of software created on the basis of neural networks technology is explained. The program developed allows the user to read an image, to create and to train a neural network, and finally, to classify the entire image. Using the implemented software it is possible to develop a unique program which processes efficiently all the data. Actually, one of the thematic principals of this material is the classification of satellite images. Given that an image acquired by the satellite could be composed by different bands not visible to the naked eye (and therefore not interpretable), an automatic program, automatically trained, is the ideal method to study the various of information contained in it.

1. INTRODUCTION

The potentialities of neural networks in processing remotely sensed imagery have been shown in several studies (for example Benediktsson, 1990; Del Frate *et al*., 1999). Neural networks have great capabilities as a pattern recognition method for multi-source remotely sensed data because of the parallel nature of the processing. In addition, no prior knowledge is needed about the statistical distribution of the classes in the data sources. In particular, it has been shown that multilayer perceptrons (MLP) may be an efficient alternative to conventional statistical approaches for automating image classification. However, the design of neural algorithms is not trivial and several critical issues have to be properly managed to build reliable procedures (Bishop, 1995). Indeed, at least four main steps are necessary for a neural network technique being implemented for image processing: *a)* generation of a statistically meaningful set of training data, *b)* definition of network topology, *c)* training phase, *d)* application of the trained net to the entire image. So far, different tools containing a neural network simulators have been released in the field of public domain or freeware software. In particular we mention the SNNS (Stuttgart Neural Network Simulator) software, which we largely have been using for our research activities. However SNNS, like most other similar software, being conceived for general applications rather than being focused on remote sensing, lets the user to interplay easily with phase *b)* and *c)* allowing, once the training files are available, to build the most suitable neural network. However, step *a)* and step *d)* are outside the simulator environment. This means that the user usually has to deal with other software environments and build his own routines both for generating the training and validation sets and to apply the designed net to the entire image, with the difficult result that everything has to be started again if the final result is not satisfactory.

In this paper we present a new tool based on neural networks for remotely sensed imagery or, generally, for image processing, which includes in the same package all the four aforementioned steps. This means that the user can easily manage the whole processing in a unique software environment. In fact, the tool, that we preliminarily call NEURANUS (NEURAl Network User Simulator), allows the user to interact with the image for the selection of the training sets, to create the network topology and perform the training algorithm, and to realize in real time or near real time (depending on the size of the image) the results produced on the base of the choices performed in the earlier steps.

2. THE TOOL

2.1 General characteristics

The tool has been developed within an IDL environment, it is based on a window (widget) interface and it is rather user-friendly because very little knowledge is required of the users about neural network theory. In fact, the tool automatically implements and executes some actions masking them to unskilled users, while others aspects can be easily handled and solved by means of a trial and error approach. As a result, the tool provides the pixel-based classification of the full image chosen as input. In figure 1 is shown a diagram that summarizes the principle choices carried out by the user, from the start to the finish of the program. We can distinguish among 3 main fields of interactions between the user and the software: the first field regards the managing of the files and the generation of the training patterns. The second field concerns the design of the neural network. The third one manages the final classification result. The corresponding functionalities will be described with more detail in the following paragraphs. From the point of view of its graphical interface, two main fields can be distinguished as illustrated in Fig. 2. A two-windows area is basically dedicated to show the input image and the current corresponding classified image provided by the net. A control panel where the user can act on the classification process by means of buttons or keyboards inputs.

2.2 From the image to the training patterns

The software accepts various types of image formats, in particular those provided by satellite imagery distributors. Once the file is selected, it is displayed on the left-hand window and the users can start selecting the regions of interest by defining the classes of interest. Figure 3 shows this functionality for a user choosing the training pixels for the class vegetation.

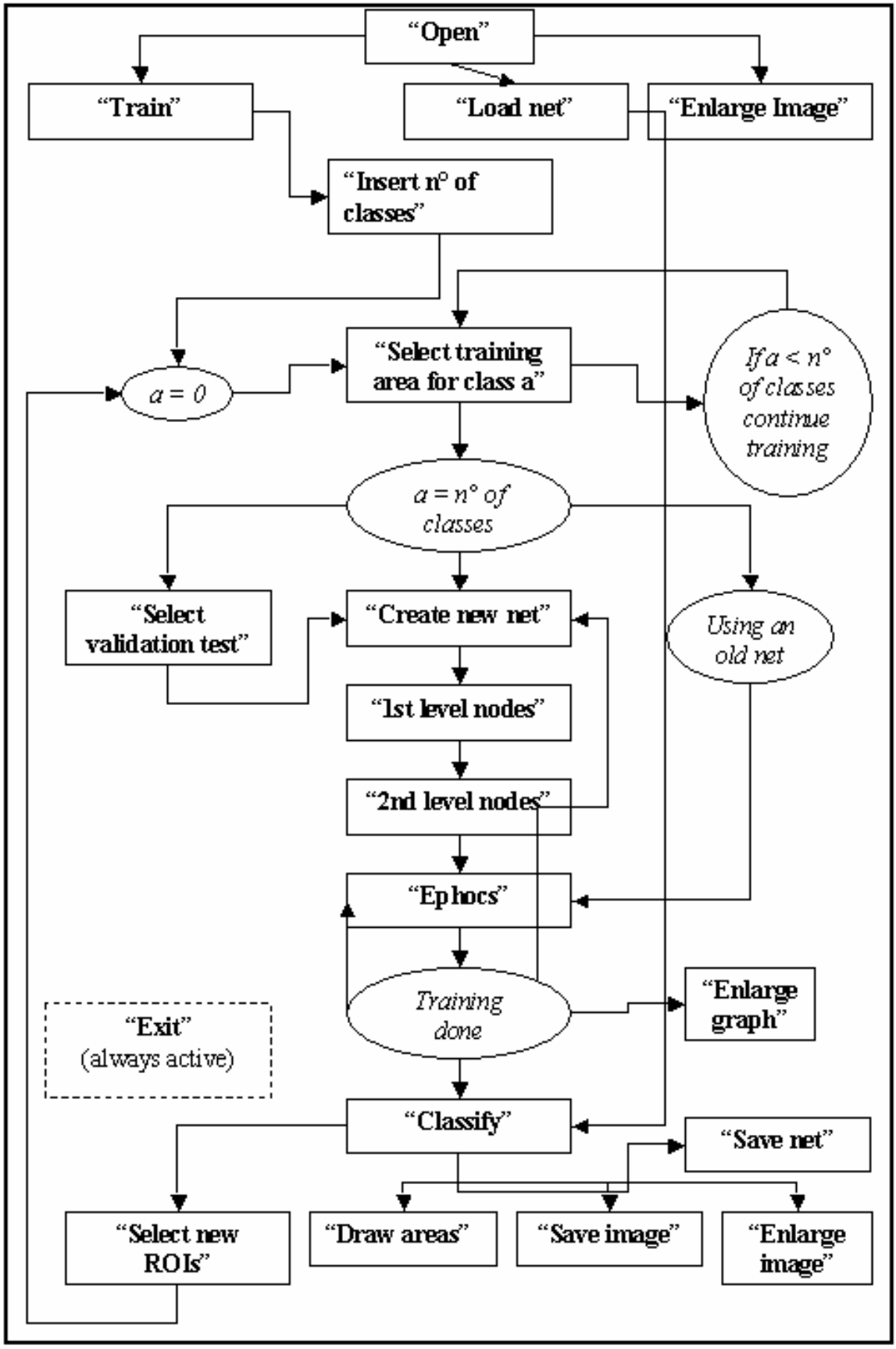

Figure 1. Block diagram illustrating the possible choices of the user

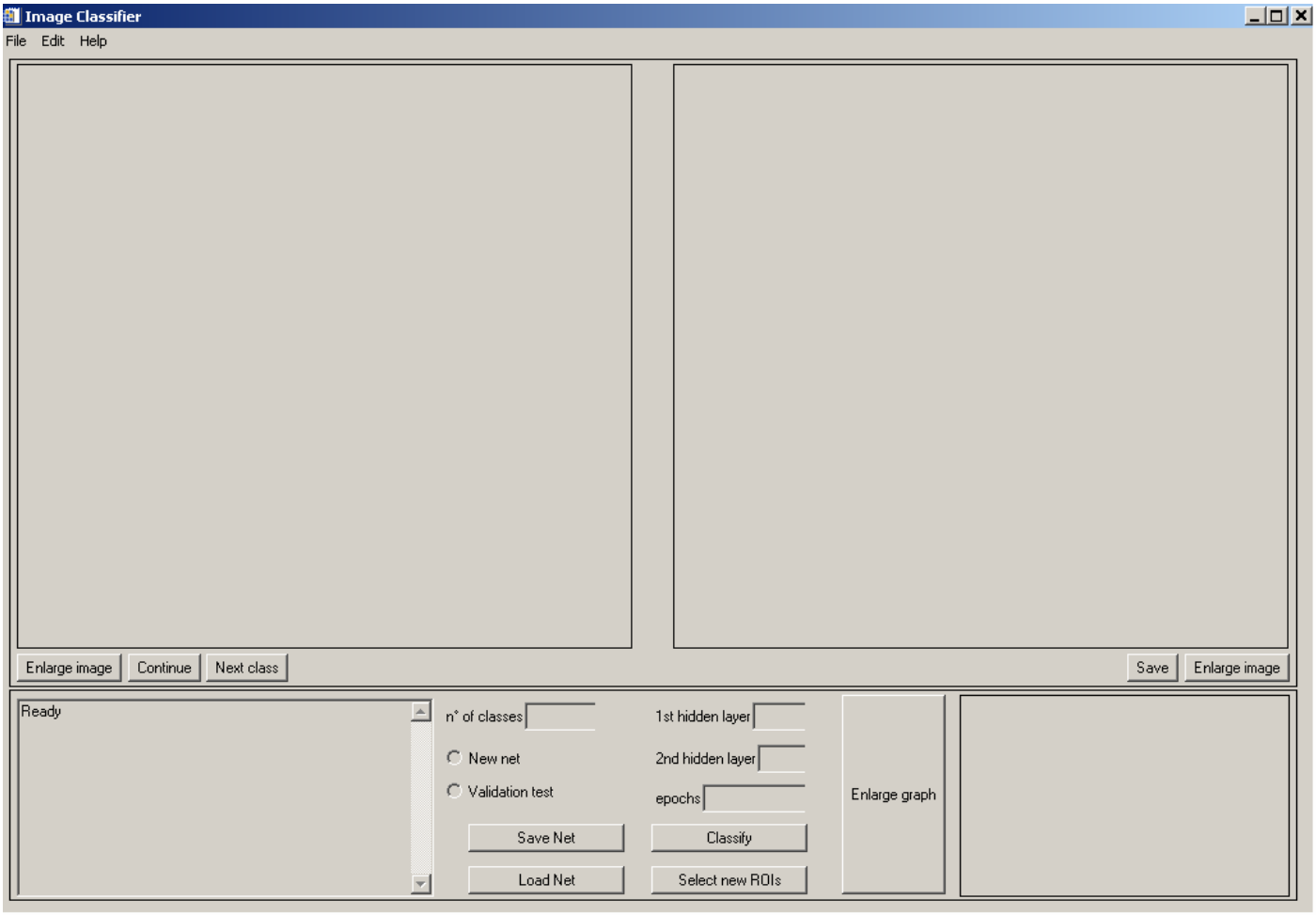

Figure 2. Principal dialogue window of "NEURANUS"

Once the area has been selected the user may choose to select another training region for that class or switch to other classes. When the selection of the training areas is closed, the tool automatically generates the training set. Before storing the final patterns, a scaling of the diverse bands is executed, in a way to obtain all of the input values in a range between [-1,+1], with (-1) associated with the minimum value and (+1) associated with the maximum of the i^{th} band. Shuffling techniques, which mix the created patterns and put them in random order, have been also included in the software to optimize the successive training phase.

2.3 The neural network

The design of a suitable neural network algorithm involves the consideration of two main issues: the definition of a proper topology for the network and the implementation of the learning algorithm. As far as the topology of the network is concerned we decided, at least for this first release of the software, to fix the MLP as the only possible architecture but to let the user to choose, in addition to the dimensionality of input and output layers, the number and the size of the hidden layers in the network. For the training, two error minimization algorithms have been considered: the standard backpropagation algorithm and a fast learning algorithm based on scaled conjugate gradient (SCG) technique. This is a member of the class of conjugate gradient methods; these are general purpose second order techniques that help to minimize goal functions of several variables. Second order means that such methods use the second

Figure 2. A training area is selected for the first class (vegetation). The desired area is the upper left red red polygon. For the other classes the procedure is the same

derivatives of the error function, while first-order techniques like standard backpropagation only use the first derivatives. The software initializes the weights from small random values. To create the net, the user inserts the number of the epoch *e*, which represents the number of times the training on the input pattern

is executed. When the training is finished, a graphic that reports the decrease of the total error as a function of the epoch number will be written in a dialogue window. Before beginning the training phase it is possible to insert or not to insert a *validation test.* This test consists of selecting some ROIs separately that are analyzed by the net during the training phase. The utility of the validation test is that it represents a good measure of the progress of the training; on the basis of the Back Propagation theory we know that, at least when finding a local minimum, the error on the training set always decreases. This does not mean that a longer training phase is more accurate, on the contrary, during a long training phase the peculiarities of the input pattern could be missed, giving more attention to the specific example with the danger of incorrectly altering the values of the connection weights of the neurons. However, using this procedure, it will be possible to obtain a very low error value in the training phase, but a not very accurate final classification. Using the validation test on the error graph, the error of the validation set, which is external to the training set, is printed. In this way the user can check the progress of the training (Fig. 4).

2.4 The output

The output consists of a classification of the image according to the classes specified by the user. The result can be accepted, rejected or improved. Leaving the same topology, the user may try to improve the results either acting on the number of learning epochs or adding new training examples. When the classification result is judged as satisfactory it is possible to save it in a output image file. A neural network is basically characterized by the values of the weights for each level; after the classification it is possible to save also the connection values obtained from the training algorithm during the learning phase. The user, after having checked by visual interpretation the results, could store the net in a directory that has as a name the date and the hour in which the net has been saved (in this way it is possible to save many nets, if it is necessary). If one wants to classify directly an image without the training phase, the user reloads a previously saved neural network. The utility of this option is that the user could save as many nets as he wants, each to classify successfully a given image. When the training phase is finished, an error log is printed on a specific file.

2.5 Running the software

We provide now an example of the use of the NEURANUS software in the sphere of the satellite image interpretation.
We want to classify a satellite image acquired by QuickBird on the basis of four principal classes: buildings, bare soil, vegetation, roads. The acquisition area is a part of the "Tor Vergata" University campus, located in Italy, South East of Rome. In addition to the buildings belonging to Tor Vergata University, private houses are also visible in the image. As input to the net we want to use the available four bands (R, G, B, IR). We extracted from the image an overall number of 700 training pixels. For the network topology we choose two hidden layers both consisting of 12 neurons. The learning was based on a simple backpropagation algorithm iterated on 900 epochs. In Fig. 5 we show the software graphic interface after the processing which lasted about 1 minute although running on very economical platforms. We see that the result is already satisfactory at this point, even if it may also be improved acting on the described functionalities.

3. CONCLUSIONS

A new software package for image classification based on neural networks has been developed. The package is able to automatically link the several steps necessary for producing results with neural networks. Among them we include the generation of significant training and validation set, the network design and the application of the net to the entire image.
The software provides its output (the classified image) in near real time so that restarting the procedure by introducing modifications in the network topology or in the training phase does not involve long time consumption. We have shown an example of application to QuickBird imagery, in particular, its 4 bands have been used. The classification performance, obtained in few minutes, confirms well the interesting capabilities of neural networks for this type of processing (Del Frate *et al.*, 2004). The software can accept (for the moment) a maximum of 20 bands so it can be considered both for more traditional missions such as Landsat and to process images provided by the latest sensors for monitoring land use on board IKONOS or Proba, or the instruments such as ASTER, which is deployed on the TERRA satellite.

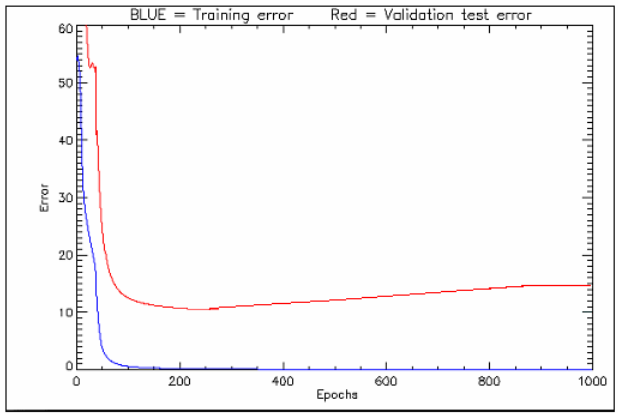

Figure 4. Graph of the error of the input patterns and of the validation set. It is printed by the program in a suitable dialogue window

REFERENCES

Benediktsson, J.A., Swain, P.H., Ersoy, O.K., 1990. Neural Network Approaches Versus statistical methods in classification of multisource remote sensing data. *IEEE Trans. Geosci Remote*, 28(4), pp. 540 - 552

Bishop, C., 1995. *Neural Networks for Pattern Recognition*, Oxford Univ. Press, New York.

Del Frate, F., J. Lichtenegger, D. Solimini, Monitoring urban areas by using ERS-SAR data and neural networks algorithms, Proceedings of the International Geoscience And Remote Sensing Symposium, Hamburg, Germany, 1999.

Del Frate F., G. Schiavon, C. Solimini, Application of neural networks algorithms to QuickBird imagery for classification and change detection of urban areas," Proceedings of International Geoscience And Remote Sensing Symposium, Anchorage,Alaska, 2004.

SNNS, www-ra.informatik.uni-tuebingen.de/SNNS/

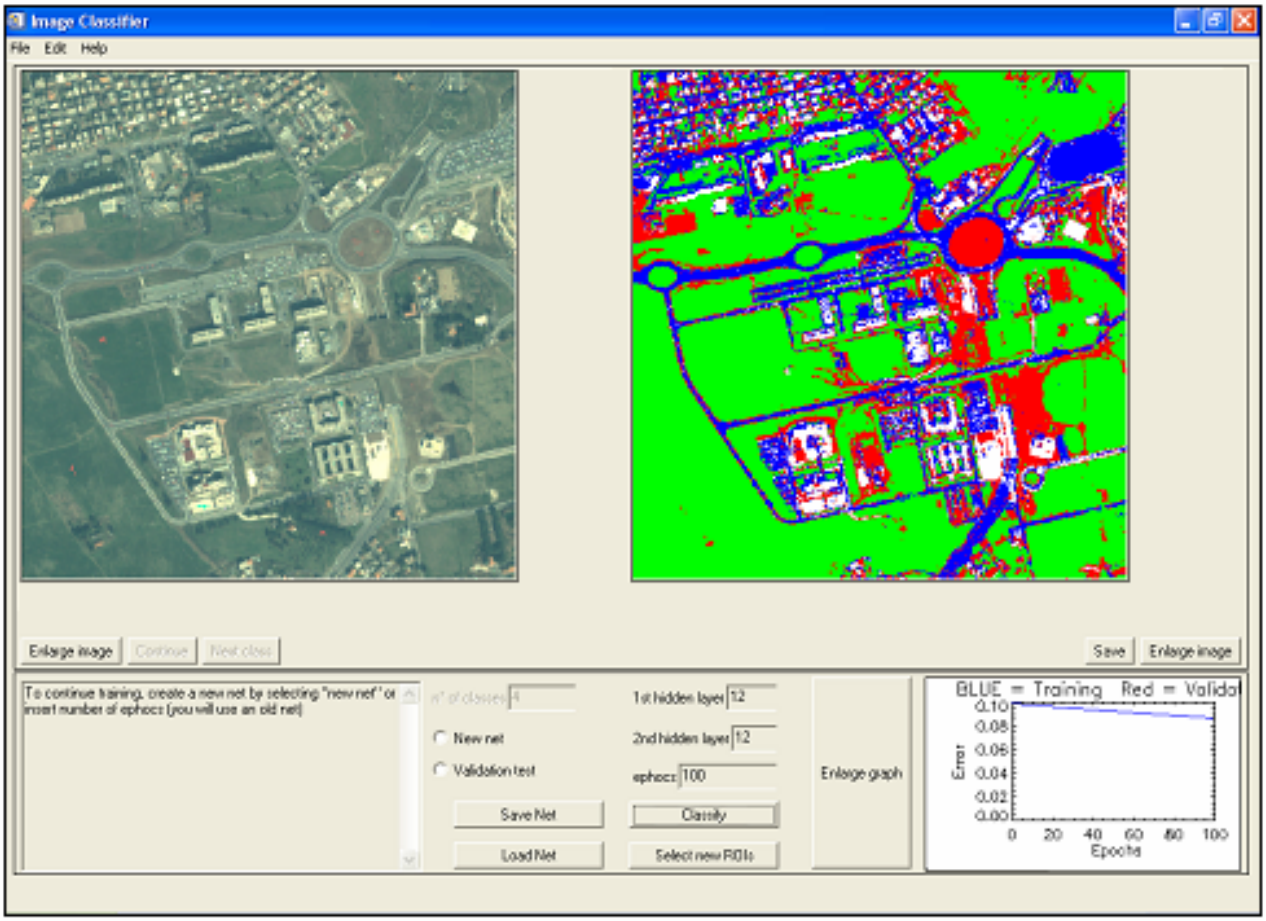

Figure 5. An accurate classification of the “Tor Vergata” University area. We have four classes: vegetation (green), bare soil (red), roads (blue) and buildings (white).