ABSTRACT: It is increasingly usual to find notions of artificial neural networks (ANNs) included in the curricula of many engineering studies. In this article, we present Neuro-Lab, a simulation software to teach basic concepts of ANNs to undergraduate students of the Computer Engineering School in the University of Vigo (Spain). Neuro-Lab was designed with the aim of offering a very easy-to-use user interface, support of exploratory learning activities, and also high reusability. This feature allows for an easy extension, integration, and inter-operation of the software with other applications. © 2003 Wiley Periodicals, Inc. Comput Appl Eng Educ 11: 93–102, 2003; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.10042

Keywords: artificial neural network; simulation software; reusability

INTRODUCTION

Artificial neural networks (ANNs) are not only an active research area, but also a tool that has proved to be valuable to solve complex problems in many different application fields of science and technology [1,2]. As a result, it is increasingly usual to find
and training phases. We think that for engineering students, the most important is to understand this dynamic nature, and the real utility and operation of neural networks. Classroom-based teaching, books, and lecture notes are not sufficient to transmit this kind of knowledge.

One way to solve this lack of accessible information problem is to allow students to do practices using computers and simulation software, which provide procedural knowledge [10]. Simulation can play a valuable role when developing these abilities in the students [11]; that is why it is the most popular kind of computer-based teaching in university education [12]. One of the reasons for including simulations in students’ training is the possibility of showing the dynamic characteristics and temporal evolution of systems [13,14], which is particularly interesting in the context of neural networks teaching. Moreover, using computers with suitable simulation software avoids having to deal with all the tedious and complex calculus underlying ANNs, thus overcoming student’s loss of interest [15]. Finally, efficiency of simulations as compared to classic training methods like theoretical lessons or laboratory practices, has long been appreciated (i.e., [16–18]).

All these arguments motivated us to introduce a computer-based learning environment with simulation software, in order to teach basic concepts of ANNs to undergraduate students of the Computer Engineering School in the University of Vigo (Spain).

Another different aspect to consider in simulation software is its integrability and reusability. In the last years these issues have gained a considerable relevance in educational software development. Some authors consider integrability and reusability as critical requirements in order to avoid the so-called “pattern of failure” of educational software [19–21]. Simulations are a kind of software where interaction and integration features are particularly valuable. For example, it could be interesting to be able to collect output data from a simulation to graphically present them or statistically process them. Also it could be useful to integrate different heterogeneous simulations as components of an open modeling environment, in turn of developing closed complex environments [21]. This is even more interesting in cases like ANNs, as they are a tool of ubiquitous application in many different areas of science and technology.

The remaining sections of this article are structured as follows: firstly, we explain the requirements we defined for the software and justify the decision of building our own software Neuro-Lab. Next, we explain the main features of Neuro-Lab and its reusability issues. Finally, we make some conclusions.

**REQUIREMENTS FOR AN ANNs SIMULATION SOFTWARE**

We defined the following requirements the software simulation tool had to fulfil according to our aims.

**Very Easy-To-Use Interface**

That is an obvious requirement, as we have serious time restrictions, and the student can not start to effectively learn after being familiarized with the interface of the environment he or she is using [15].

**Support for Exploratory Learning**

We aimed at providing a constructivist learning computer-based environment, basing on a simulation software to support discovery learning processes [22]. Therefore, it is preferred to be a highly visual and graphical interface supporting direct manipulation of parameters and network components in an intuitive way [23].

**Support for Extendibility, Interoperation, and Integration With Other Applications**

The software has to run in the platform available for laboratories practices, which is PC-Windows. It should also be able to easily interact with other applications commonly used by students, particularly worksheets. Furthermore, it should also be desirable that the software could interact or be integrable with other software, specially with modeling and simulation software. It is also important that this software be easily extended to support new features, as new training algorithms, or new kinds of ANNs. These features can be resumed in the requirement for the software of being highly reusable.

Many powerful neural network simulators are available both in the market or on the web. Outstanding examples are SNNS [24], PDP++ [25], Xerion [26], or the Neural Networks Toolbox for Matlab [27]. Despite the excellent quality of these software tools, they are excessively complex to use for an undergraduate student, specially when the main goal is to transmit a very general picture about ANNs, centered on a small number of well-defined concepts, and do some clarifying, brief and simple practices on the computer.
There also exist many other ANNs software simulators less powerful and more simple to use. For example, in Reference [9] we can find a catalog of several Java software tools oriented to education in ANNs, which are available on the web. However, most of the existing tools are clearly oriented to illustrate a very concrete application or kind of ANNs, allowing limited or no interactivity, neither exploration of the internal behavior of networks.

There also exist several general-purpose simulators (i.e., BrainWave [28], NeuroSolutions [29]), some of them offering a wide range of possibilities to create arbitrary networks, manipulate them, and show results. The interface of some of these simulators allows for a more or less direct and visual manipulation of networks and their parameters. But in most of the cases we found limited capabilities of interoperability and integration with other software tools.

Most of the educational software is not explicitly developed to be integrable neither reusable, which seriously hinders its potential reusability [20]. Even if a programming language a priori well-suited for reusability is utilized, for example Java, this is not enough at all to guarantee reusability of the developed software [30,31]. Reusability, which implies other desirable features like interoperability or integrability, has to be explicitly considered when developing and delivering software.

All those considerations lead us to the decision of building a ANNs software simulation tool, called Neuro-Lab, with the aim of fulfilling the requirements described above and paying particular attention to reusability issues.

**NEURO-LAB MAIN FEATURES**

Neuro-Lab allows to build single-layer and multilayer neural networks, thus for example, perceptron or backpropagation networks (BPN). Currently support for Self-Organization Maps (SOM) and Radial Basis Function (RBF) networks is going to be added.

Networks in Neuro-Lab are built in a very intuitive and visual manner. The user first selects the kind of network and its number of layers. Subsequently the layers are displayed as color-filled and labeled empty shapes on the workspace (Fig. 1). When the user clicks over one of these shapes, a dialog box asks him to introduce the number of neurons to create in the layer, and their default activation function (e.g., sigmoid, hyperbolic, or linear).

Each activation function has a different color associated with it, to visually recognize the function of a neuron. As Neuro-Lab allows creating heterogeneous layers where neurons can have different transfer functions, this feature is particularly visual and intuitive (Fig. 2). Each neuron also displays its actual value at its upper-left corner, and its identification at its bottom-right corner. This identification is a pair composed of the layer number (beginning from zero for the first layer) and the order number of the neuron in the layer (also beginning from zero). This

![Back Propagation Networks. Neural networks design software.](image)

**Figure 1** Empty layers of a artificial neural network (ANN) shown as colored shapes in Neuro-Lab.
Identification system is used all along the application, for example, to show the origin of input and output connections of a neuron (Fig. 3).

Once the neuron set for a layer created, the user can modify it, adding or deleting neurons, or changing the activation function of a particular neuron as well as of all the layer. All these operations can be made by clicking directly on neurons with the mouse to display the neuron-editor dialog box (Fig. 3), which also permits to see/modify the output value of any neuron as well as of the weights of its input/output connections.

The weight of the connections between neurons also follows a color code, which legend can be displayed as an emergent window (Fig. 2). This allows the student to visually realize weight updates during the training stage of a network.

As this is the most common, the neurons of each layer are by default totally connected to those of the previous and next layers, but this option can be deactivated directly in the layer-creation dialog box. The user can add or delete any connection simply by activating the connection edition mode and clicking on the desired neurons.

Once completed the network design, the user can add some training or testing data, pasting them from the clipboard (thus from a worksheet, a database, etc.), importing them from a standard text file or simply using the integrated random generation of input data. Neuro-Lab allows user to create data pages, which are named grids, which can contain data. The user can select the active data page for a particular operation. This permits easily managing multiple datasets for training/testing a network.

Once introduced at least a dataset, we can train the network, selecting between incremental or batch training the learning rate and the number of options. It is also possible to display and adjust the shape of the activation functions. Thus we provide the student with an easy and graphical way to observe and understand the consequences of a parameter modification [32]. For example, in Figure 4 we can see the steepness increase of a sigmoid activation function, which is

\[ f(x) = \frac{1}{1 + e^{-Gx}} \]

when increasing the value of \( G \) from 4 to 8.

The evolution of the output and error rate can be graphically displayed during a training cycle. It is also possible to visually observe the changes of the weights of connections through their color code. Once we consider a network to be sufficiently trained, we can test it with the same or another dataset, simply by selecting the desired data page, and observe its output either graphically or numerically. The correlation coefficient between the expected and real output data series is also provided.

Neuro-Lab supports copy and paste of data from/to the clipboard or a text file. Thus input data can come from quite any other application and output data can be exported to any application for its analysis,
post-processing, more complex graphical representation, etc. Users can also save and retrieve a neural network to/from file.

As the steps of a typical ANN life cycle are commonly building—training—validation—exploitation, in this order, Neuro-Lab toolbars and menus are organized in the same way, thus offering the items for each one of these step consecutively ordered.

As it can be seen, Neuro-Lab’s user interface has been carefully designed. We think it suitably fulfils the
defined requirements, particularly with regard to interface easiness and support for exploratory learning. The assessment of the students’ response to the software environment, collected by means of questionnaires, seems to confirm this opinion. Thus, 90.3% of them assert that they appropriate the utilization of a software tool to teach neural networks basics. Also the 92.4% considered that the software is very friendly and easy to use.

**REUSABILITY FEATURES OF NEURO-LAB**

As pointed out, one of the requirements for Neuro-Lab was achieving a high degree of extendibility, integrability, and interoperability with other software. All those three features are deeply related to reusability of software, as reusable software must be integrable in other systems, extensible to adapt it to new requirements, and interact with other software in an easy way [31].

The best way to fulfil this requirement was promoting reusable features in Neuro-Lab’s design and implementation. The most promising approximation for software reusability is to adopt a component-based development [30,33]. This involves, in one hand, developing software as a composition of reusable components, and in the other hand developing new software to be reusable. Another way to improve reusability at design level is to use design patterns [34] to build the software model. It is also important to use standards at the different levels of software development to ease its integration and extension [31].

Basing on these considerations, it was decided to use an object-oriented approach to build the software. Even if object orientation is not enough to assure reusability, it has been widely accepted that it favors it, as software is developed as a set of classes with a clearly defined client interface, but hiding implementation details [30,31,35]. Neuro-Lab class hierarchy was designed using the standard UML object-oriented modeling language, and implemented using an object-oriented programming language.

For example, in Figure 5 we can see the class diagram to model an ANN in Neuro-Lab. Thanks to using abstract classes and the factory pattern [34], the model allows easy extendibility by simply adding new kinds of ANNs, as well as new training algorithms or activation functions.

As previously pointed out, interoperability is achieved, in its simpler way, by providing copy and

---

**Figure 5** Class diagram of an ANN in Neuro-Lab.
paste functions from/to the clipboard and export/import support from a text file.

More complex interoperability and integration with other software has been achieved by firstly decoupling user interface, which can hinder reusability of a software component [31], from internal ANN model, following the Model-View-Controller (MVC) architecture. This allowed for directly exporting the class hierarchy of the ANN as a component library, both as DLL and ActiveX, which are currently de-facto standard component architectures for Windows-based platforms. Including the capability of retrieving ANN models from file in the component library, we can design and training a network in Neuro-Lab, save it in a file, and afterwards loading and using it into another application which uses the component library. This permitted, for example, to integrate Neuro-Lab with Model-Lab, a modeling and simulation tool able to import DLL components [36]. Thus, it is possible to include ANN built in Neuro-Lab into models built in Model-Lab.

The Neuro-Lab ActiveX component also allows for integration of Neuro-Lab in web-based applications. This is a very important feature as web is gaining huge relevance in delivering educational software [37]. In fact we recently developed a web version of Neuro-Lab reusing the ActiveX component, and also reusing almost all the user interface code of the original application, changing only some aspects of to adapt them to the web features (Fig. 6).

Another feature we found potentially interesting in order to enhance Neuro-Lab integrability was make possible the easy exchange of ANN models with other applications. To allow this, it is important to use standard platform-independent file formats. Therefore, XML was selected as the format for saving Neuro-Lab’s ANN files (Fig. 7). As XML is becoming the standard for information exchange, we can expect this feature allowing for interoperability with the increasingly common XML-capable applications, and even indirectly with other non-XML compatible applications thanks to the suite of tools for translating XML data to other formats.

CONCLUSIONS

In this article, we have introduced Neuro-Lab, a simulation software developed at the Informatics Department of the University of Vigo, in order to teach ANNs. We have exposed the main features of the software, emphasizing those oriented to support a more easy, intuitive and exploratory use of the software environment.

We also have explained the most relevant aspects of the software design and implementation that
provide to it a high degree of reusability. Using object-oriented design and programming languages, design patterns, allowed us to ease Neuro-Lab extendibility. Compliance with component-based standard architectures made available integrating ANNs into other modeling software as well as into web applications. Finally, supporting easy data copy and paste and using standard formats as XML for saving information enhances the interoperability of the application with other software.

We think that developing Neuro-Lab software to be highly reusable has thus proved to be very useful in many aspects, and corroborates the increasing attention paid to this concern in educational software development.

REFERENCES


B. Choi and E. Gennaro, The effectiveness of using computer simulated experiments on junior high school students understanding of the volume displacement concept, J Res Sci Teach 24 (1987), 539–552.


Emilio García Roselló is lecturer at the Department of Informatics of the University of Vigo. His research work is mainly focused on reusability and component-oriented software engineering, particularly in their applications in educational software. He can be reached at erosello@uvigo.es.

Jacinto González Dacosta is lecturer at the Department of Informatics of the University of Vigo. He has been working for many years in educational software development. His recent research work is on component-based technology and learning objects. He can be reached at jdacosta@uvigo.es.
J. Baltasar García Pérez-Schofield is lecturer at the Department of Informatics of the University of Vigo. His research work is mainly focused on object persistence and component-oriented software engineering. He can be reached at jbgarcia@uvigo.es.

Manuel Pérez-Cota is chair professor at the Department of Informatics of the University of Vigo. He heads research groups on educational software and object-oriented software engineering and he has many publications about these issues. He can be reached at mpcota@uvigo.es.