# MyANN: An Educational Software Package About Artificial Neural Networks

*Taivo Lints*

Tallinn University of Technology
Department of Computer Control
Ehitajate tee 5, 19086 Tallinn
ESTONIA
taivo@vkg.werro.ee
http://www.dcc.ttu.ee/taivo/

## ABSTRACT

*This paper describes a small free open source software package developed by the author and called MyANN. The package is created to facilitate people's understanding of artificial neural networks (ANNs). It can be used as the first introduction to ANNs or, due to its distinctive network visualization capabilities, also as a debugging tool for ANN specialists.*

## 1. INTRODUCTION

There are many software tools available for creating and using artificial neural networks. However, most of them are not very suitable for getting a quick and easy overview of the processes going on inside the net. This is a serious drawback, especially for the students just wanting to get acquainted with ANNs, but sometimes also for specialists trying to debug their nets.

The lack of attractive, enjoyable, simple, educative, free and preferably open source programs about artificial neural networks motivated me to try to create one such software package myself, mostly because I had no prior experience with ANNs and I wanted to get a good overview of them. The alpha version of MyANN package was finished in 2003. Due to several reasons the project was then put on hold for two years and I have not advertised and distributed it during this time because of its alpha version status. However, the development will now continue and as this package, especially the component called Visualizer, seems to have quite a good educational potential, it might be a good idea to start introducing it to a wider audience.

This paper is organized as follows: in section 2 the MyANN package is described, one program per subsection, and section 3 concludes the paper.

## 2. MyANN PACKAGE

MyANN is a small educational software package about artificial neural networks and it contains three small programs: Trainer, Visualizer and LightChaser. With these programs it is possible to create networks, visualize their inner workings and try out an interactive demonstration of their practical use. In addition to the compiled programs MyANN also contains a "soft" introductory text about ANNs and quite heavily commented C++ source code.

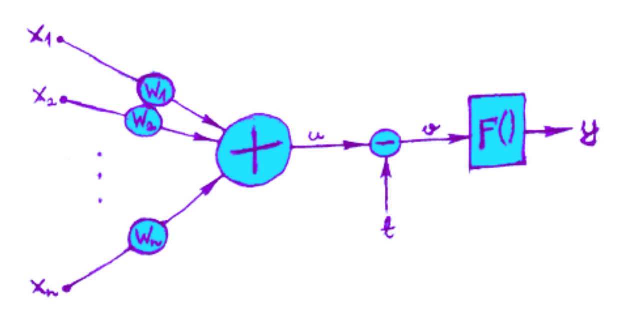

**Fig. 1.** MyANN currently uses this typical structure of an artificial neuron. On the left there are inputs $x_1, x_2, \ldots, x_n$. Each input goes through a connection, which has a certain weight w, and is multiplied by that weight: $x \cdot w$. All these multiplied values are then added together: $u = x_1w_1 + x_2w_2 + \ldots + x_nw_n$. This sum is then adjusted with threshold t, producing $v = u - t$. And finally a so called "activation function" F() is applied to the adjusted sum, giving us the output of the neuron: $y = F(v)$. MyANN uses sigmoidal activation function, specifically the "logistic function".

Currently only multilayer feedforward nets are implemented and neurons have the typical structure and functionality as shown in Fig. 1.

### 2.1. Trainer

Trainer is a simple tool without graphical user interface (maybe it will be added later) for creating a multilayer feedforward network and training it with backpropagation algorithm. It uses two configuration files and produces one output file, all being human readable text files. In one of the files users can specify network structure (the number of layers, and the number of neurons in each layer) and neuron parameters (the threshold value, and the slope parameter for the sigmoidal activation function).

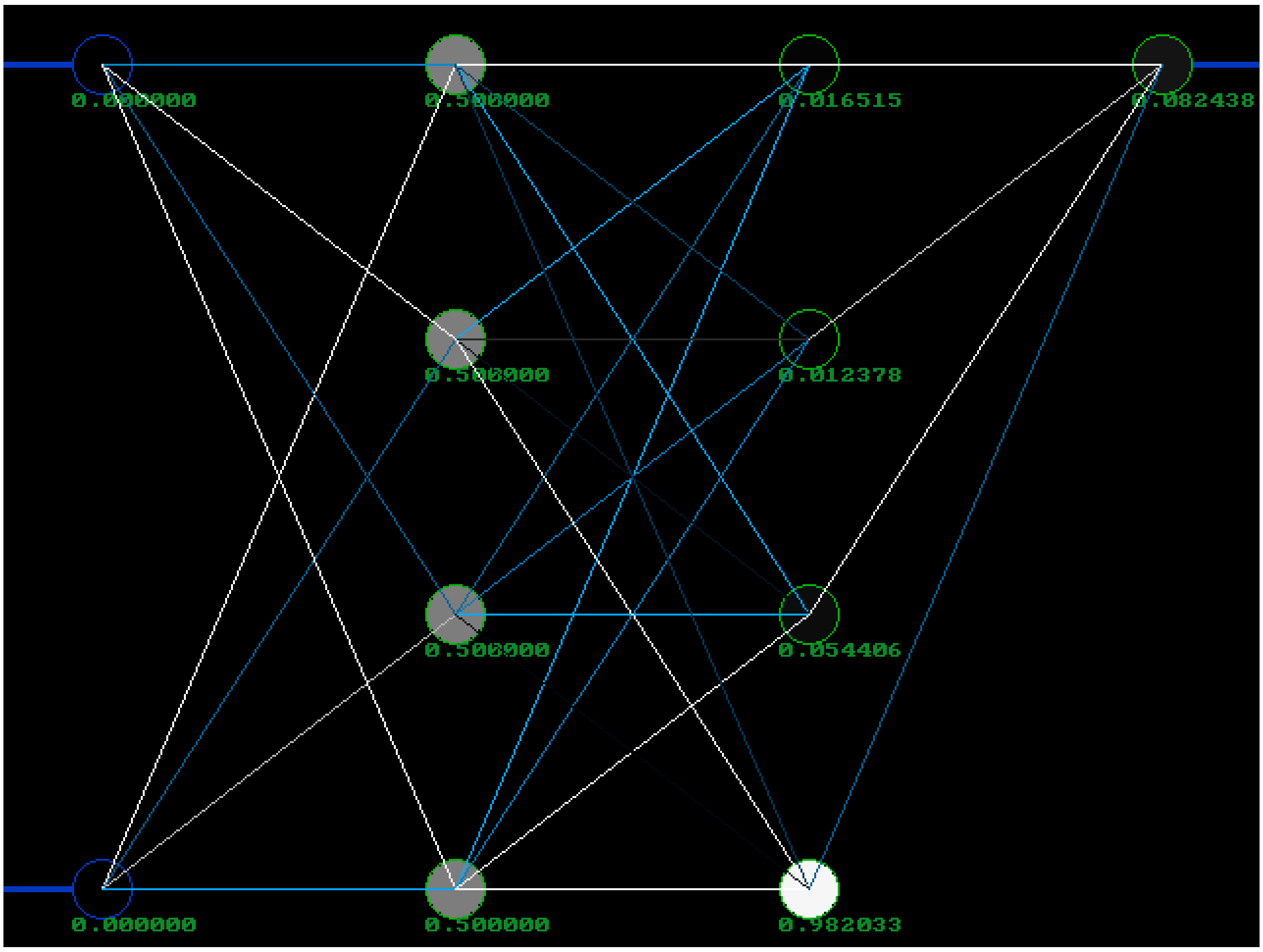

**Fig. 2.** A screenshot of the Visualizer. Blue bold lines on the left edge are input pins, similar lines on the right are output pins (here only one output). Blue circles on the input pins are input nodes that distribute input values into the network. Green circles are neurons, filled with a color that can vary from black to white through shades of grey, indicating the output value of that node (black is 0, white is 1). Connections between neurons can vary from brightest blue (the most negative allowed weight) through black (weight 0, not visible, i.e. no connection) to white (the most positive allowed weight).

The other file contains training patterns and related parameters, such as error limit (for a very simple evaluation of the success of training: if all outputs are in a range from (desired_output – error_limit) to (desired_output + error_limit), then training is completed), learning rate, the maximum number of training iterations and the maximum allowed weight for a connection.

The result is saved into a text file that can be used as a configuration file for the following two applications.

## 2.2. Visualizer

Visualizer, the most distinctive component of this package, allows users to see what happens inside an artificial neural network. The ANN is displayed on screen and information about important values – the weight of every connection and the output of every neuron – is given using color coding (see Fig. 2).

What makes Visualizer different from other similar software, however, is the way users can interact with the program. The "ease of play" has been the main aim here, which translates into the ease of changing the ANN's input values and getting immediate response. To change the input value on the first input pin smoothly up or down between 0 and 1, the user has to hold down key "1" on the keyboard and move the mouse up or down, respectively (pressing mouse buttons is not necessary). Key "2" corresponds to the second input, etc. Discontinuous changes in input values can be generated by first moving the mouse, then pressing the key on keyboard. Changing several inputs simultaneously is possible by holding down more than one key at the same time, though then all corresponding inputs have the same value and due to the keyboard hardware some key combinations may not work. All changes in inputs immediately spread through the network, changing the output values of the neurons – and therefore their colors on screen – and giving the user an insight into the ANN's inner processes that translate the changes in inputs to changes in outputs.

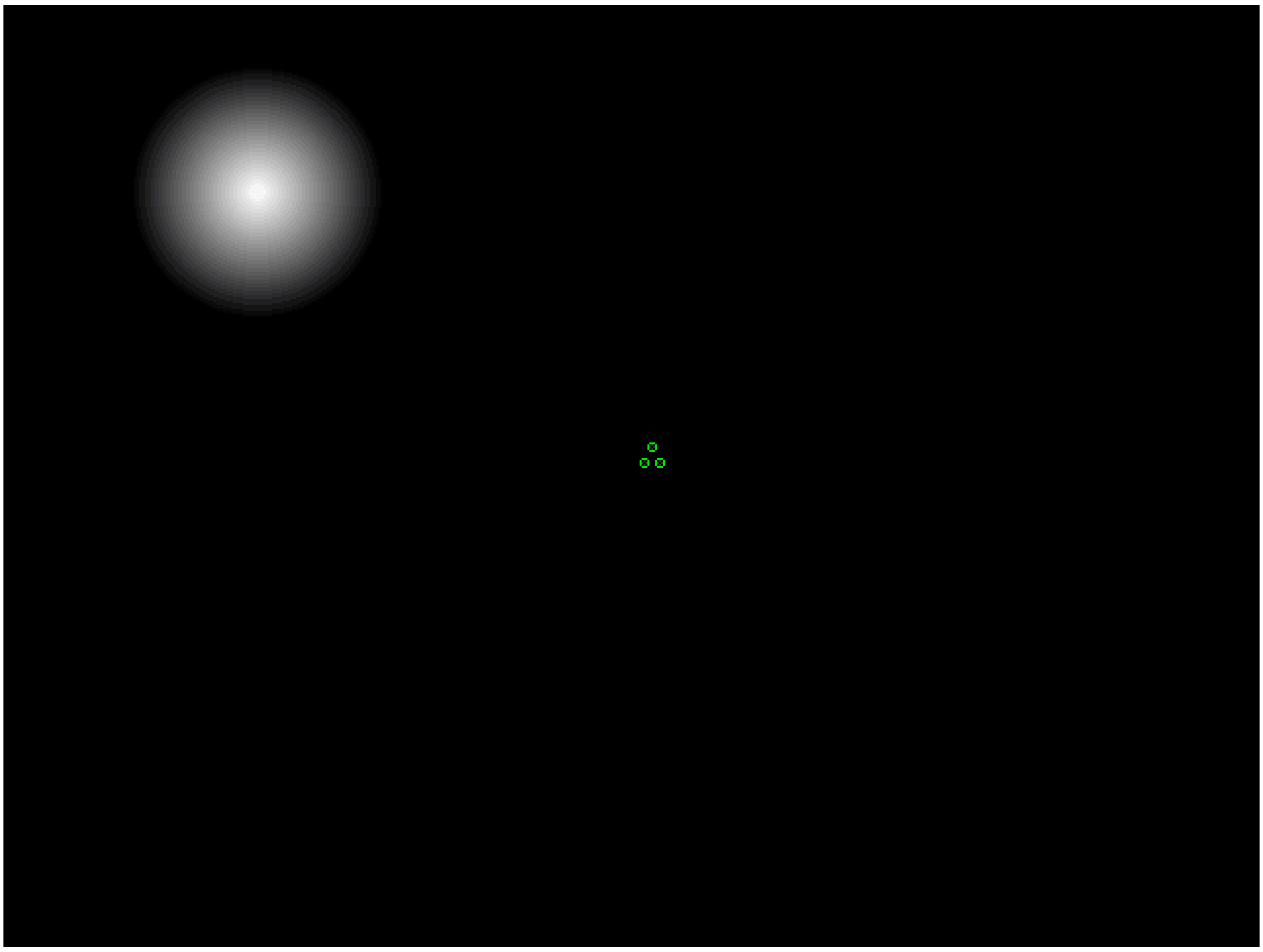

**Fig. 3.** A screenshot of LightChaser. The "creature" with its three sensors can be seen in the center and a moving spot of light in the upper left corner.

### 2.3. LightChaser

LightChaser can be used for demonstrating the practical use of artificial neural networks. It simulates a light sensitive "creature" (or robot) whose movement is controlled by an ANN.

The playfield (see Fig. 3) contains the creature and a moving spotlight. The creature has three light sensors. Each sensor reads the light intensity exactly from the center of the sensor and feeds it into the ANN. Outputs of the neural net determine creature's movement, for example the creature may try to move away when the edge of the light "touches" it. A new ANN can be loaded on the fly. The speed and direction of light's movement can be changed, or the light can be moved manually with mouse. The creature can also be dragged manually if necessary.

This example may not be a very practical one, but hopefully can serve its purpose: to keep up students' interest after the inevitable question: "And what are these ANNs good for?".

## 3. CONCLUSION

The software package MyANN seems to be quite useful, mostly for getting acquainted with artificial neural networks (currently multilayer feedforward nets only), but in some cases also the specialists might find it helpful to "play" with their nets using the Visualizer. MyANN is still in alpha stage and needs further development, but everybody is already welcome to visit project's webpage http://www.dcc.ttu.ee/taivo/myann/ where more detailed information can be found and the current version of the package can be downloaded.