A General Display for the Visualisation of Artificial Neurons

T.D. Gedeon & J. Lin

School of Computer Science and Engineering
The University of New South Wales
P.O. Box 1, Kensington 2033, Australia

1. ABSTRACT

Building successful neural networks is still largely a black art, depending on a great deal of specialised experience. Information presented to users is very voluminous, being the weights and activations of all the neurons, and any expert has to be able to extract the relevant information. By data visualisation we can manipulate the data so that the less experienced can more easily process the data to reach the same end. We present a simple animated colour display of neurons and their interconnection in a neural network which can be used with any standard network model. We show that by our visualisation technique finding redundant neurons becomes simple, which otherwise requires significant computation to perform automatically, or requires great expertise.

2. VISUALISATION OF NEURONS

In a neural network, there can be a large number of neurons, classified as input, hidden, and output neurons; for some models, there is no such distinction. The neurons could be arranged into a single layer, or into multiple layers; into pools without layers; or into pools within different layers. These neurons could be fully interconnected, including self-connection, or some arbitrary subset of this. The connections could be bi- or uni-directional, and can be excitatory or inhibitory.

When we study a particular neural network model, we are interested not only in its topological structure, but also its weights and activation and their changes, net input, output, goodness, and error. In some cases, we are interested in the ratio of internal and external input as well.

In many cases, after the topology of network is set, we are mainly interested in the computation of each neuron. When a neural network is in use, the weights on connections between neurons are fixed and error does not change. We are interested in how the combinations of different weighted and directed connections plus external inputs and biases affect the activation levels and their changes, and thus form the competition, blocking, resonance, hysteresis, etc., thus the internal state of the neural network. We could show the error surface such as in [1], but this does not show how the network finds a minimum, worse it provides no information on what has gone wrong at the individual neuron level if it is trapped in a local minimum. During neural network training, we are interested in how weights change, how the error reduces, how the internal state changes when more patterns are learned. To show the internal state of a neural net is a high dimensional scientific data visualisation problem.

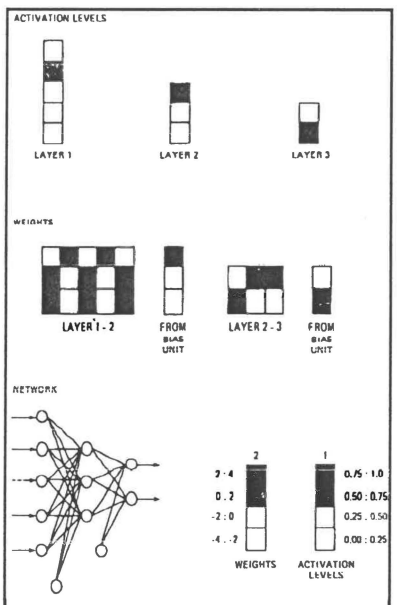

Figure 1. Graphical representation of activation levels and weights.

A common display is that of Figure 1 [2]. There is a simple mapping between components so figuring out which activation level box corresponds to which neuron is possible. Nevertheless, interpreting the diagrams meaningfully is difficult in that we must constantly match parts of the diagram. It is not clear from this diagram whether any of the neurons are redundant or similar to each other on this input pattern. As the target pattern is not given, we assume the network was trained and is in use.

While it is possible to become adept at discerning meaningful patterns in such displays, or from the standard voluminous numeric displays, these all require significant time to learn.

To take advantage of the powerful information processing ability of our eyes and the visual processing parts of our brains, we developed a method to display the interstate of neural network which has complex, dynamic features as described above. We describe the general technique, and then specialise the display topology for the back-propagation network we will use in our examples.

3. THE "FLOWER" METHOD

We develop an information box for each neuron in a neural network. The box contains graphic information about the activation, input (including external input, internal input, bias, net input, and internal input ratio), output, equilibrium/ excitation /inhibition indicator, selected weights, and so on. We arrange these boxes into a circle regardless of the original topological structure so that we can concentrate on the behaviour of the units. This arrangement can be modified for particular models where the topology provides contextual aids for the user (as for back-propagation), or where the topology is meaningful, for example in the regularity detection paradigm.

Some design considerations:

Activation is one of the most important properties which we want to visualise as easily and accurately as possible. We are interested in its value, its changes, and in comparing between neurons. Usually the activation is in a range [Min, Max], so we design the activation as a bar in the box. The height of the box represents the range, and the length is proportional to the value. Red is the chosen colour because it is one of the most 'visible' colours in nature. Thus, we can easily and accurately capture and compare the dynamic feature of activation by the animation of the activation bar. The figures shown here are produced with a alternative black/white mapping.

One exception to the range idea of activations is in linear auto-associator model: the activation can be any real number. Here we re-scale the boxes if some activation becomes too large.

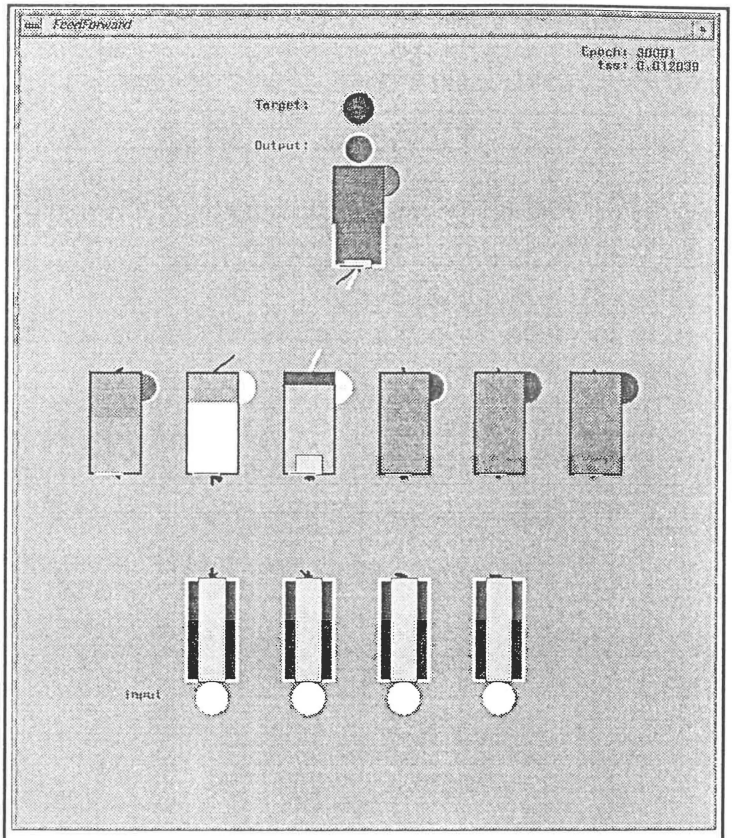

Figure 3a. Visualisation of feed-forward back-propagation network using specialised display. (B/W diagram)

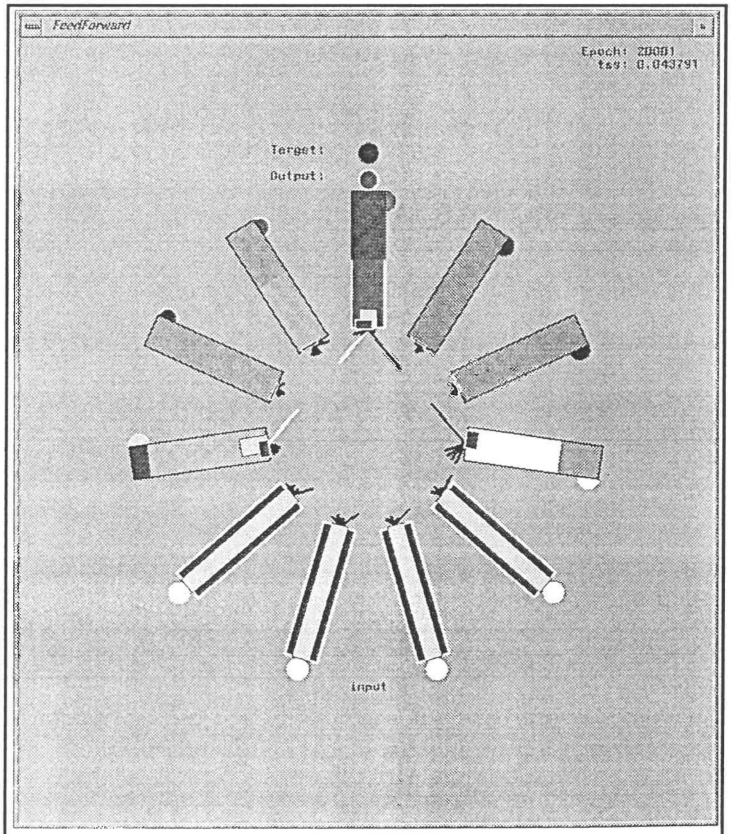

Figure 3b. Visualisation of feed-forward back-propagation network using general "flower" display. (B/W diagram)

Inputs (internal, external, bias, net), output could be any number. It is not necessary to visualise them accurately, but is enough to know if there is any input or output, and if an input is stronger or weaker than others, if the output is similar to the target, etc. Thus we use colour, density to display them.

The weights are the most difficult to visualise. There could be many connections in a network, and their weights could be any real number. There could be 4 kinds of connections (uni versus bi-directional, positive or negative) between neurons. Without a well chosen method to display the connections and their weights, we will produce a useless mess. When a network is in use, the connections and their weights are fixed, so different users may want to see different groups of connections. When the network is in training, we want to see all the weights because the network is adjusting each of them so it is not possible to decide on a meaningful subset to display.

According to all of these reasons and constraints, we arrange the information boxes in circle, design the connections to be colour bars, and display these colour bars in the middle of the circle. When the network is in use, the user can select the range of weights to be displayed, this defaults to a range larger than the largest current weight to allow visible growth on screen without requiring rescaling. When the network is being trained, at the very beginning all weights would be small random values close to *0*, so there are no colour bars are visible on the screen due to their shortness. Thus we display significant weights, without cluttering the screen. The magnitude of the weights is represented by the length of the colour bar, and its sign by colour. During training, each weight is increasing: we display this as two small bars enlarging proportionally to its value towards each other from either end of the connection for bi-directional weights. For unidirectional weights, the colour bars extend from the source towards the destination neuron only.

Absolute weight magnitude is more important than the sign. We use colour to represent positive & negative signs.

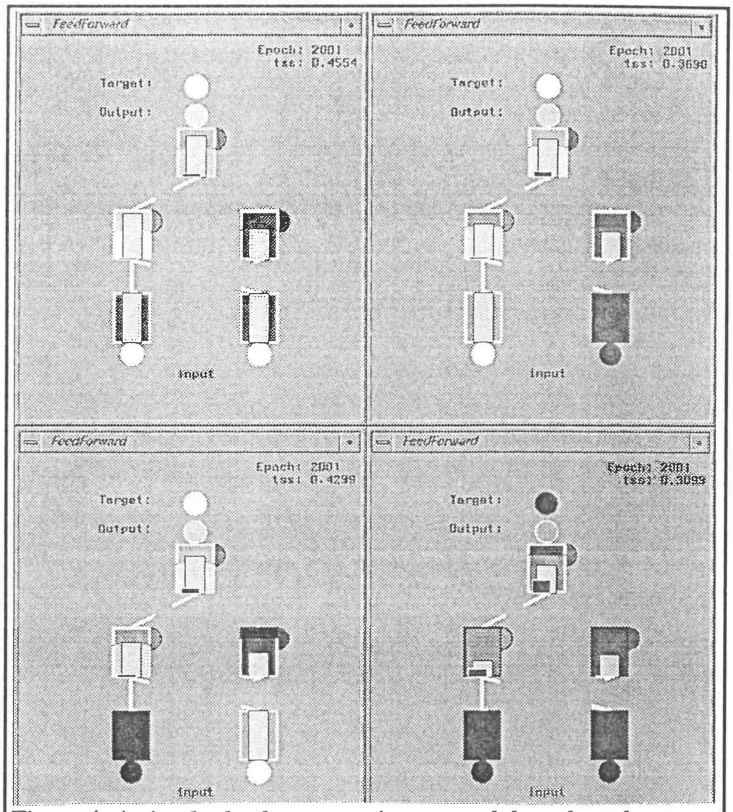

Figure 4. A simpler back-propagation network by selected pattern (B/W diagram)

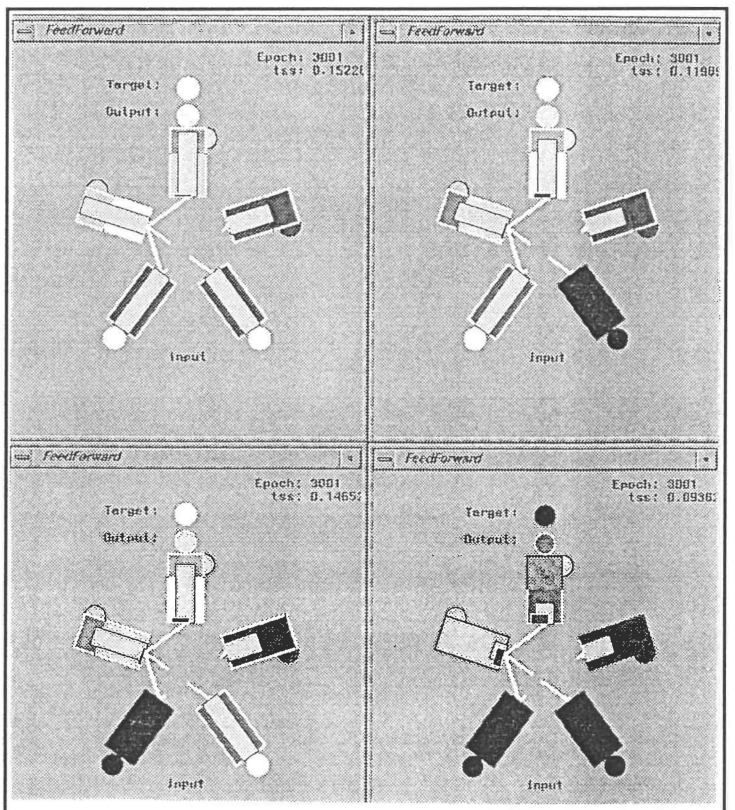

Figure 5. A simpler back-propagation network by selected pattern using "flower" display. (B/W diagram)

Green is used for positive, while blue is used for negative, with black being zero. This is consistent over all of the weights, biases, and the internal / external net input displays. For weights, the magnitude is important and is encoded by the length of the colour bar. For other properties using our display, the magnitudes are less significant and are displayed as the shade of colour. Thus a bright green semi-circle shows a high positive bias.

For connections to the outside world, values are assumed to be normalised to the range from *0* to *1*, being the almost ubiquitous practice. Values in this range are represented by shades of colour filling the input, output or target circles from cyan representing *1* to black representing *0*.

In the back-propagation network, we are interested in the error at each neuron. This is represented by a yellow bar of the same shape as the red activation bar, but partially overlapping it. This choice of location gives a number of visual cues. If the error is high, the actual activation value is not significant as the network will need to learn that pattern better. Once the error is low, the red activation bar is no longer even partially occluded as its value is now more meaningful.

A back-propagation network discussed and analysed previously [3] is shown in Figure 3. It is clear from a glance, that at least for this input pattern the activations of neurons 1, 4, 5 and 6 (counting from the left in the middle layer) are very similar. Since the weights do not change for each pattern presentation as significantly as does the activation, the small size of the weigths also indicates that these four neurons are very similar. The rightmost 3 neurons could be removed without affecting the processing of this network. The recognition of this was trivial compared with the sort of computation which is otherwise required and validates the usefulness of our method of display. The "flower" display at a later epoch shows that only two neurons are significant.

A simpler network is displayed in Figure 4, again using the layout for back-propagation. A real-time display with a single window is not useful for showing a network during training on a number of patterns, as the activations and error change noticeably, and not particularly meaningfully between patterns. Instead, a number of windows are used each showing the effects of the same pattern in successive epochs. This produces a seamless animation automatically. The number of patterns displayed should be small, and chosen to be representative. It is known that most of the time learning is spent on a subset of the patterns [4], allowing reasonable choices.

The same network is displayed in Figure 5, demonstrating the generic "flower" display. Note that one of the neurons is redundant, having very little effect on the output neuron. The "flower" display does not detract from the context of layers, as we can place the hidden neurons between the input neurons at the bottom and the output neurons at the top. The internal input ratio is shown by the relative size of the two regions of back-ground colour in the neurons. In the top right quadrant, the top neuron's (green) background demonstrates the ratio of internal input to the sum of the absolute values of the inputs. In effect, the size of the upper box indicates the significance of the internal input, and the shade from blue through black to green its sign and magnitude.

4. CONCLUSION

The general display mode can be used for any standard neural network model. We have already considered the following models: back-propagation, including cascaded and recurrent networks; interactive activation and competition networks, including Kohonen feature maps; pattern associator and auto-associator models. The animation of all these weights, as well as of the activations, discloses the internal state of the neural network which is very difficult if not impossible to comprehend without this visualisation.

We have demonstrated that using an animated colour display of the internal behaviour of a neural network during training, we can easily see when a neuron is redundant and is not learning. This is a significant result which otherwise requires some computational effort.

REFERENCES

1. Perry, J, "Basins of Attraction Revisited," *IJCNN*, Seattle, vol. 1, pp. 853-857, 1990.
2. Dayhoff, J, *Neural Network Architectures*, Van Nostrand Reinhold, 1990.
3. Gedeon, TD, Harris, D, "Network Reduction Techniques," *Proc. Int. Conf. on Neural Networks Methodologies and Applications*, San Diego, vol. 2, pp. 25-34, 1991.
4. Slade, P & Gedeon, TD "Bimodal Distribution Removal," *Proc. IWANN Int. Conf. on Neural Networks*, Barcelona, 1993.