Artificial Biochemical Networks: A different Connectionist paradigm

CHRISTOPHER MACLEOD and NICCOLO F. CAPANNI

School of Engineering, The Robert Gordon University, Aberdeen, UK
(Email: chris.macleod@rgu.ac.uk)

Abstract. Connectionist models are usually based on Artificial Neural Networks. However, there is another route towards Parallel Distributed Processing. This is by considering the origins of the intelligence displayed by the single celled organisms known as Protoctists. Such intelligence arises by means of the biochemical interactions within the animal. An artificial model of this might therefore be termed an Artificial Biochemical Network or ABN. This paper describes the attributes of such networks and illustrates their abilities in Pattern Recognition problems and in generating Time-Varying signals of a type which can be used in many control tasks. The flexibility of the system is explained using legged robots as an example. The networks are trained using Back Propagation and Evolutionary Algorithms such as Genetic Algorithms.

1. Introduction

Artificial Neural Networks (ANNs) are well-known connectionist models based on the operation and behaviour of biological neurons. Such networks have proved useful in several fields including Pattern Recognition and Control. However, they also have several disadvantages, which are outlined later in this paper.

The system presented here is an alternative connectionist model. Just as Biological Neural Networks produce complex behaviours in multicellular animals, this alternative model is also biologically inspired, in this case by the complex behaviours of single-celled animals (O'Shea, 2005). Similarly, just as the Neural Network is based on the interactions between neurons - which results in multicellular intelligence, this new model based on the interaction between proteins - which produces single-celled intelligence. Such a network may therefore be termed an Artificial Biochemical Network (ABN).

The basic principle behind the network was first proposed by Capanni and MacLeod et al (2005); in this paper it is expanded and results are presented which illustrate its flexibility. In particular, the paper demonstrates that the network facilitates the simple control of motors and other actuators, as well as pattern recognition. The simplicity with which it achieves this, gives it an important advantage over other models in practical systems.

To understand the model it is first necessary to review its inspiration and origins. These are presented in the next sections.

2. Single-Celled Intelligence

Protoctists are remarkable organisms. They consist of nothing more than a single cell. There are many species, the largest of which is just visible to the naked eye - the smallest require microscopes to see (Curtis, 1969). This is a range of sizes greater then the relative size difference between a Blue Whale and hen. These which behave like animals are known as Protozoa, these which photosynthesise like plants are known as algae and the fungi-like ones, moulds. Many display traits of all three groups - for example, being able to both photosynthesise and hunt for food. One which is familiar to many people is the archetypal amoeba *Amoeba proteus,* that often lives at the bottom of ponds.

Protocists display an incredible variety of behaviours, lifestyles and habitats (Alberts et al, 1994). Many hunt actively for food, following “scent trails” laid down by their prey. Some “walk” on leg-like appendages. Still others have light sensitive spots which can act like “eyes” or are sensitive to vibrations with delicate cilia “ears.” A few not only hunt for food but have poison “darts” which they can shoot out at their quarry to paralyse it before consumption. And some even build themselves shelters in which to hide. Truly such organisms display many of the traits of intelligence.

One must remember in reading the paragraph above, that we are talking here about Single-Celled Organisms. There is no neural network or any other similar system directing their behaviour - the neuron is itself, after all, a single cell. The complex and interesting behaviour described above comes from another source - the biochemical reactions which control the cell. How this comes about is considered in the next section.

3. Biological and Artificial Biochemical Networks

The complex and interesting behaviours displayed by single cells occur because of interactions within the cell, between proteins. Proteins are the true workhorses of the cell. It is proteins which the cell's DNA code specifies as is shown in figure 1. The system shown in the figure is so fundamental to life (both single-celled and multicellular) that it is often referred to as the *Central Dogma* of biology (Alberts et al, 1994).

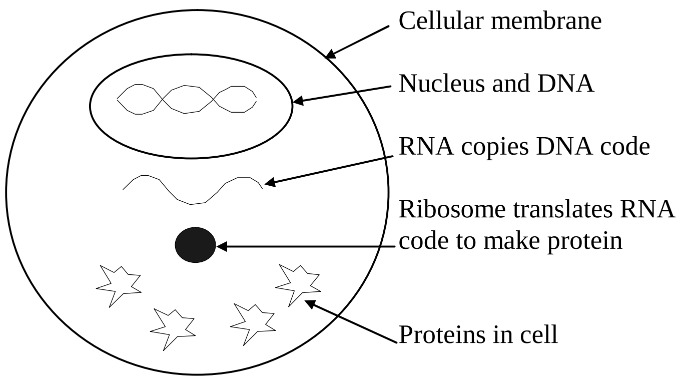


Figure 1. The Central Dogma of Biology - how the DNA code is used to make the proteins which run the cell.

Proteins perform all the important functions within the cell - they make new material and destroy old (such chemical processing proteins are called Enzymes), they sense and signal changes in both the cell's internal environment and its surroundings, and can join together to make more complex structures. They do this by binding to each other and to other chemicals. Importantly, when they interact with each other, they can form complex signal processing networks in a similar way to a neural network. This is best illustrated by a stylised example.

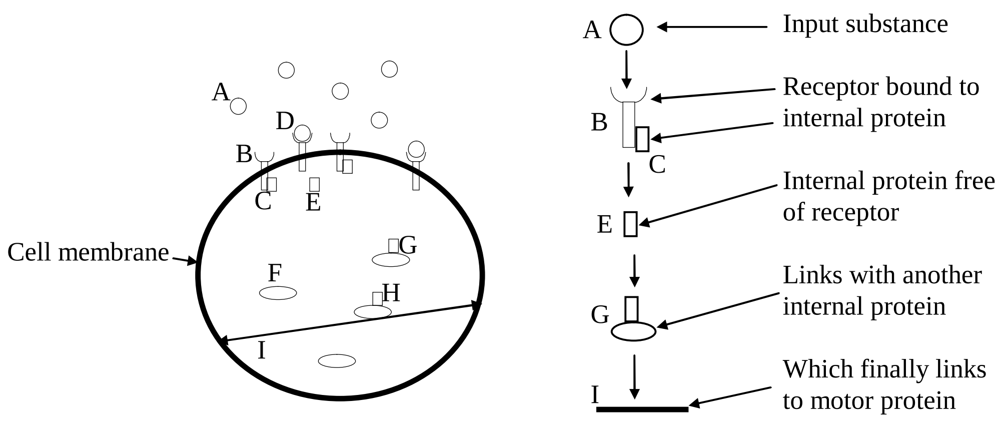

A
D
B
C
E
Cell membrane
G
F
H
I
A
Input substance
B
Receptor bound to internal protein
C
E
Internal protein free of receptor
G
Links with another internal protein
I
Which finally links to motor protein

Figure 2. An imaginary cell displaying a protein network. The proteins detect an input (the chemical "A") and through a series of interactions produce an output (a movement of the cell).

Consider a cell, as shown in figure 2. Outside the cell are molecules of a particular substance **A** (for example, a food substance). Specialised proteins on the cell surface **B** (known as receptors), which cross the cell membrane and are half-in, half-out of the cell, are shaped in such a way that they can "mate" with **A**. The internal part of the receptors is bound to another protein as at **C.** When **A** and **B** link, as shown at **D**, this internal protein, **E** detaches from the receptor (this usually happens because the receptor changes shape). All the free-floating proteins, like **E**, move around inside the cell because they are being buffeted by thermodynamic forces (similar to the way pollen grains are buffeted in Browian Motion). This free-floating protein eventually meets another internal protein **F**, which is shaped to "mate" with it, it then links as shown at **G**. This linked unit is shaped so that it can interact with a motor protein **I** as at **H**. Motor proteins (a well known example of which is Actin) when stimulated in this way contract and can move the cell - in this case (say) towards the stimulus **A**.

So a network of interactions exists between the input at **A** and the output at **I**. This network is similar in concept to a neural network, but as will be shown, has some advantages. Of course it may be represented in a stylised network diagram as shown in figure 3 (Capanni, 2006).

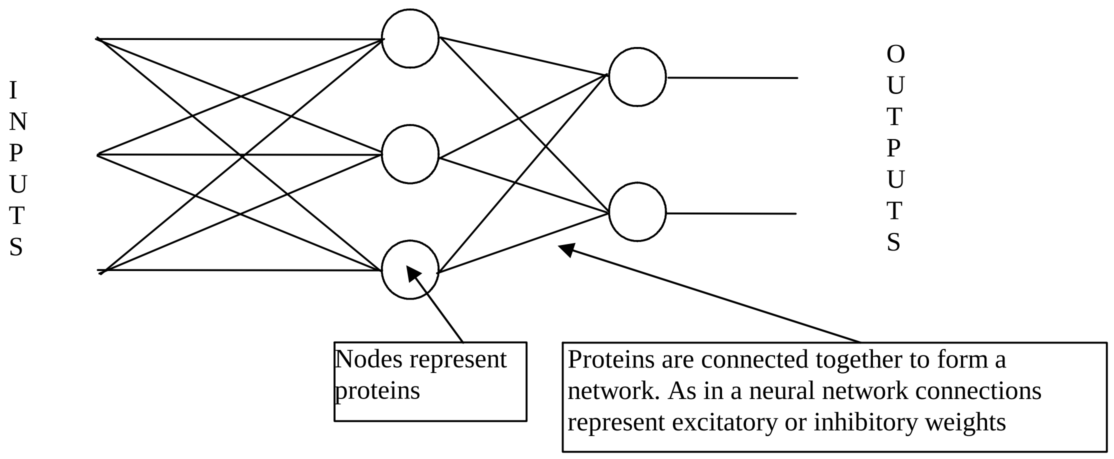

INPUTS
OUTPUTS
Nodes represent proteins
Proteins are connected together to form a network. As in a neural network connections represent excitatory or inhibitory weights

Figure 3. The Biochemical network shown as a Connectionist network.

4. Operation of a basic ABN

If the basic network is as shown in figure 3, with nodes representing proteins and weights (which are simply positive or negative floating-point numbers, as in an ANN) representing their interactions, then the next question is how do the nodes behave. Figure 4 below shows two possible behaviours.

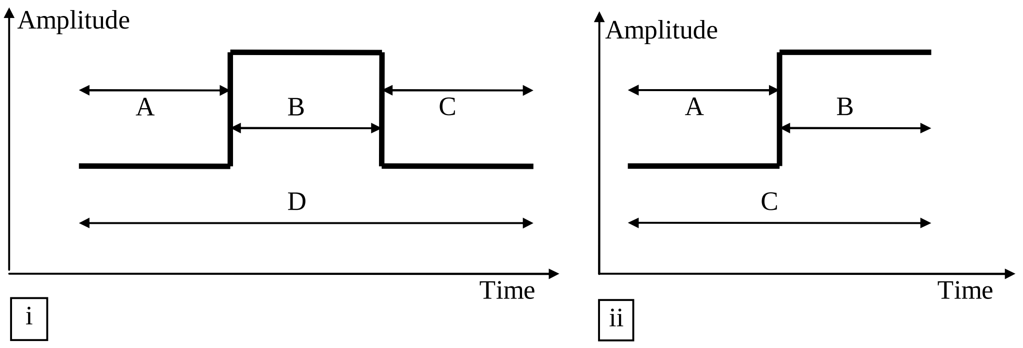


Figure 4. Each node produces an pulsed output.

In diagram i, time period A is the lag-time from stimulation of the node by an incoming signal, to the protein appearing. Period B is the time during which the protein is present (the amount of protein present denoted by the amplitude). Period C is the time required after protein production before the node can be re-triggered again. Period D is the total cycle time for the node.

It is possible to simplify this scheme by combining the “off” times (A and C in diagram i) into a single period. This is shown as A in diagram ii. The reason for showing two diagrams will become apparent in the next section - however, for the moment, we will only consider diagram i.

It is possible to train a network containing such nodes (like the network shown in figure 3) like an Neural Network. Periods A and C of each node and the weights of the network can be fixed using a Genetic Algorithm and the Period B then depends of the activation of the unit. One simple way to calculate the activation is to use a Weighted Sum and Leaky Integrator as shown in the equation below:

$$A_t = \sum_{n=1}^{m} i_n w_n + \alpha A_{t-1}$$

Where A_t is the activation at the present time (t), i_n is the nth input to the node, w_n is the nth weight of m total inputs and weights (as in an ANN). A_{t-1} is the activation at the previous time step and α is a constant between 0 and 1 which may be set by the Genetic Algorithm. The Leaky Integrator (Gurney, 1997) term (αA_{t-1}), is widely used in pulsing networks, to compensate for the absence of input for time periods during the system.

The length of time period B in figure 4, diagram i, can then be set using a simple relationship:

$$B = \beta A_t$$

Where β is another evolvable parameter. Generally, once the node has triggered and started its cycle, it cannot be interrupted and re-triggered until it has finished.

Such networks (in this simple form) have been programmed and trained using a Genetic Algorithm in Pattern Recognition, Control and Waveform Generation tasks. However, before going on to discuss results and how such networks may be used, it is worth pausing to consider how their operation and application may be made more universal.

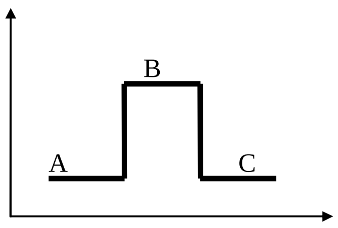
5. Analysing and Extending the basic ABN

The diagram shown in figure 4 opens up another interesting possibility. Dropping the whole idea of biologically plausibility for a moment, let us adopt an engineering viewpoint. If a network similar to this were trained using, for example, an Evolutionary Algorithm, then it would be possible to choose which of the parameters - A, B, C or D, varies with the activation of the node. Such a unit might then be more flexible (Capanni, 2006).

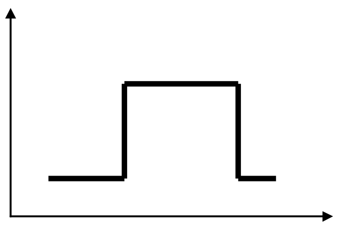
Consider that, in such a situation, each of the parameters (A, B, C and D) might have one of three behaviours (chosen by the EA). Firstly, the parameter might vary with the unit's activation. Secondly, it might have a fixed value (chosen by the Evolutionary Algorithm). Finally, it might have a complementary value - that is, it might reduce as the parameter which varies with the unit's activation increases (so leaving the total time D constant).

If such reasoning is applied to figure 4, diagram i, then the unit may have one of 81 different behaviours. However, a great number of these are contradictory or nonsensical, and cannot be fulfilled - for example, A, B and C being of fixed duration and D being variable. The others all fall into three basic categories. These are: Firstly, units which behave in a Pulse-Width Modulated way; Secondly, units which behave in a Pulse-Frequency Modulated manner; finally, those which behave as Pulse-Position Modulated units. Examples of all three of these are shown in figure 5.

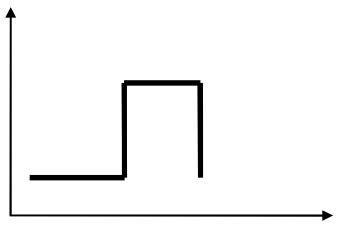
Figure 5. Examples of the three type of behaviour which can be exhibited by the unit shown in figure 4, diagram i.



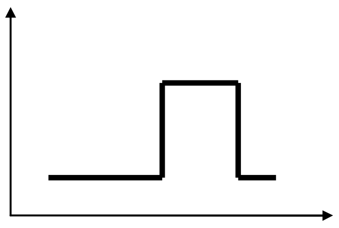
Standard pulse. A, B and C are all the same length



Pulse width modulated neuron. A is a fixed value, B varies as the activation of neuron and C is complimentary to B



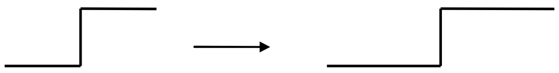
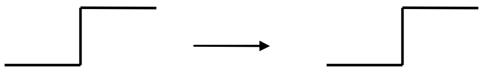
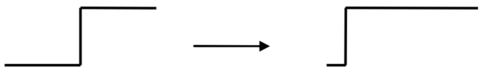
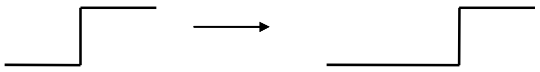
Pulse frequency modulated neuron. A varies as the activation of the neuron, B is fixed and C is fixed (equal to zero in this case).



Pulse position modulated neuron. A varies as the activation of the neuron, B is fixed and C is complimentary to A.

In fact, a unit with only three parameters, as shown in figure 4 diagram ii, can achieve all these behaviours with the exception of the Pulse-Position Modulated ones. Here there are a possible 27 different behaviours; but again, many of these are not consistent and can be ignored. The remaining behaviours are shown in figure 6.

Figure 6. The behaviours which the unit in figure 4, diagram ii can display.

	A and B fixed, C can vary - entire pulse stretches.
	A, B and C fixed - invariable pulse. A sub-category are pulses which are always high and always low.
	Pulse-Width Modulated pulse. In this case B varies with activation, A is complementary and C fixed. To get the opposite effect, swap A and B
	Pulse Frequency Modulated pulse. In this case A varies with activation, B is fixed and D varies with activation. Two other possibilities are to fix the off period, rather then the on and to let the variable parameters be the inverse of the activation (frequency goes up with activation, rather then down).

Referring to the periods A, B and C shown in figure 4 (2), it is straightforward to put mathematical detail behind these behaviours. In each case the activation is given by eqn 1 and the parameters are weighted by a constant as shown in eqn 2.

Case (i): The length C of the pulse is given by the weighted activation $C = \beta A_t$ where β and the mark to space ratio A/B are parameters evolved in the Evolutionary Algorithm.

Case (ii): This is similar to case (i) except that C is a parameter fixed by the Evolutionary Algorithm and the pulse is not effected by activation but simply repeats continuously.

Case (iii): Here $B = \beta A_t$ and $A = C - B$. The upper value of B is hard limited to C (if $\beta A_t \geq C$, then $B = C$, $A = 0$). In the case of this pulse, β and C are evolved.

Case (iv): In the final case, $A = \beta A_t$ and B is fixed period which may be either one unit or fixed by the Evolutionary Algorithm. The total period C is variable $C = A + B$.

6. The practical motivation behind the ABN

It may seem inefficient to imbue a network with all the parameters described above, when an alternative such as a Multilayer Perceptron or Radial Basis network functions with fewer. However, the motivation behind the ABN is not simply to show that there are alternative paradigms to neural connectionism, it is also because the ABN offers additional functionality.

Biological neurons produce time-varying signals in the form of Action Potentials - in contrast to the amplitude coded response of a Perceptron. This is considered important at several levels. Firstly, many researchers consider that such time-dependent behaviour is significant in the neural processes which ultimately produce consciousness (Hopkin, 1996). Secondly and more importantly from a practical engineering point of view, the configuration of certain systems can be made much more simply with a time-varying output. Example of such systems include DC motor controllers. These normally use a pulse-width modulated signal and are difficult to implement using McCulloch-Pitts or Perceptron type units as such networks require potentially unstable feedback loops to produce such behaviour.

The advantages of such time-varying units has led workers to produce several biologically plausible “Spiking” models (Gerstner, 2002). However, these are difficult and complex to implement and program and further more cannot produce the range of outputs (like pulse-width responses) required. One of the principle advantages of this new method is its simplicity and elegance in contrast with these alternatives.

7. Some Illustrative Results

In the sections below some illustrative results are reported showing the ABN in both Pattern Recognition mode (where it’s acting as an “input” network - recognising and processing patterns) and in Waveform Generation Mode (acting as an “output” network - generating patterns for use in various actuators). One of the major benefits of the network is that it can act as either or both of these simply - and is therefore a “universal solution.” Such flexibility is particular useful in Evolutionary and Robotic systems where the ability to perform in any place in the network saves the programmer having to design different unit types or resort to other technologies when the network cannot handle particular functions.

7.1 Network Setup

In the examples below, the ABN is used in both Pattern Recognition and Control tasks. To illustrate its versatility, the emphasis is on showing how both Pulse-Width and Pulse-Frequency units, of the type described above, can perform such functions. Of course, the ABN working in its most flexible form may adopt either or both of these in its structure. The unit activation is calculated as shown previously. In the case of some inputs - for example, some images, it is necessary to convert the pixel values (which are in grey scale format) into suitable pulses to be fed into the networks. This may also be applied to other inputs, for example from sensors and is done as shown in figure 7 for a Pulse-Width Modulated case.

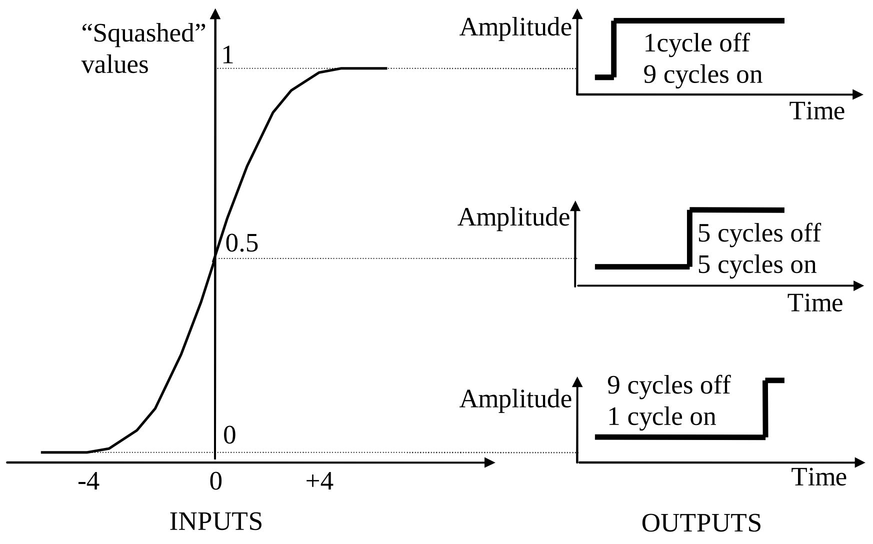


Figure 7. Converting analogue inputs to pulse trains.

In this scheme, the inputs are feed into a Sigmoidal "Squashing" function:

$$Sq = \frac{1}{1+e^{-in}}$$

Where *in* is the input and *sq* is the "squashed" function.

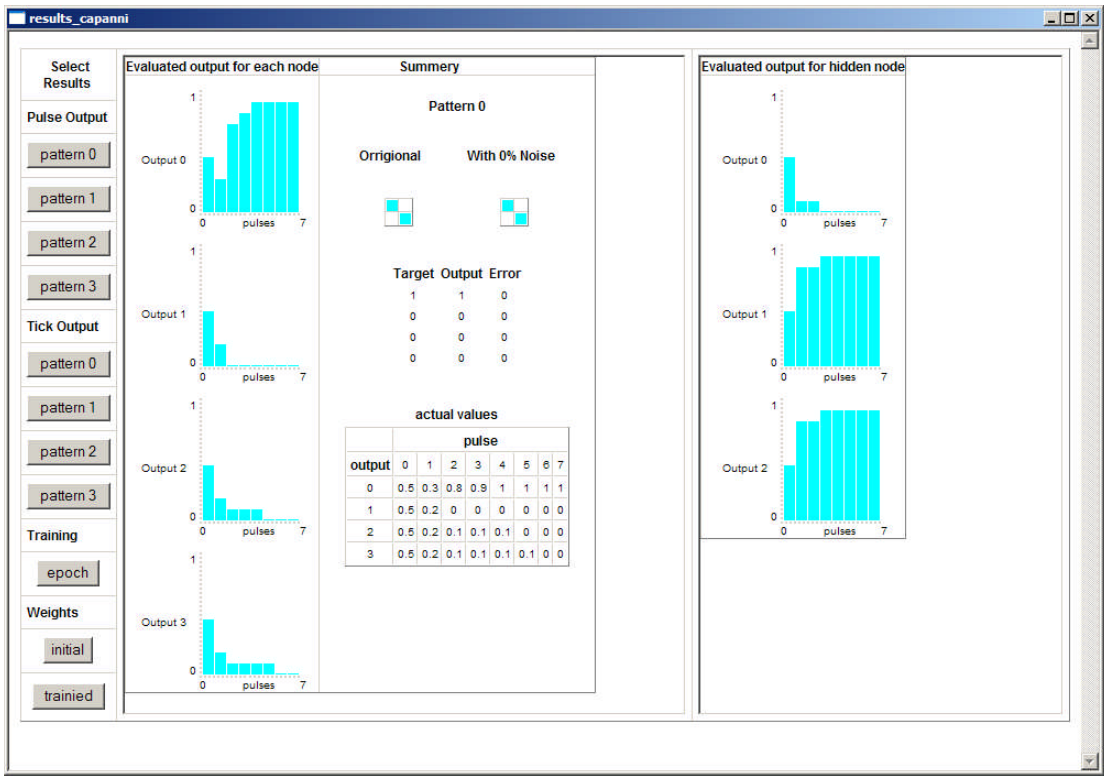
The output from this function is between 0 and 1 and is converted into a repeating pulse as shown. Value 0 corresponds to a pulse with a duty cycle of 10%, value 1 to a pulse of duty cycle 90%, the other pulses are in-between. The same system can be used for the Pulse-Frequency case. The graph may also be used in reverse - that is, to read out a pulse train and convert it to a simple number. It may also be used internally, within the unit, to convert a summed input to a pulse output instead of using the term β in the previous formulae (although this is not the method which was used to obtain these results).

7.2 Pattern recognition problems

To test the pattern recognition abilities of the network and provide a good comparison with a traditional Multi-Layer Perceptron (MLP), the problem was kept as simple as possible. The network was trained to recognise four, four pixel patterns. Having four inputs, four outputs (one for each pattern) and three hidden layer units. An MLP of exactly the same topology was also set up.

Networks containing only Pulse-Width and only Pulse-Frequency units were tested to establish whether there was any difference in performance. The Pulse-Width network was also trained using unmodified Back-Propagation to see whether such training was feasible. In this case the Target waveform was subtracted from the output waveform, time slice by time slice and the resulting errors added up over 100 time steps to give a total error for that output. Figure 8 shows the program interface for the Back-Propagation set up; the others were similar.

Figure 8. Screenshot of Program interface for Pulse-Width Back-Propagation Network.



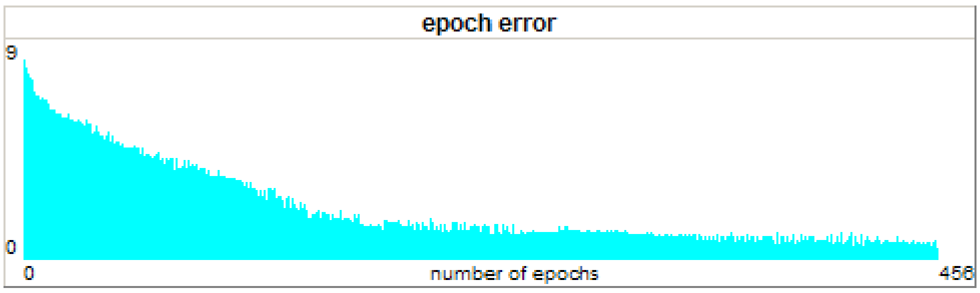
Target	Output	Error
1	1	0
0	0	0
0	0	0
0	0	0

actual values

	pulse							
output	0	1	2	3	4	5	6	7
0	0.5	0.3	0.8	0.9	1	1	1	1
1	0.5	0.2	0	0	0	0	0	0
2	0.5	0.2	0.1	0.1	0.1	0	0	0
3	0.5	0.2	0.1	0.1	0.1	0.1	0	0

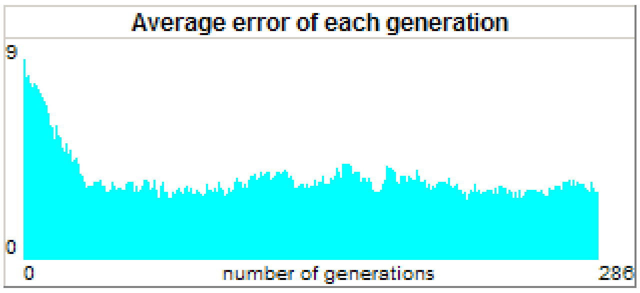
The error profile for the Back Propagation Network is shown in figure 9. It may be seen that the network trained in 496 cycles. The network was trained to an absolute error of 0.05 and easily recognised all the patterns.

Figure 9. Error verses Epoch for a Back-Propagation trained Pulse-Width Modulated Network.



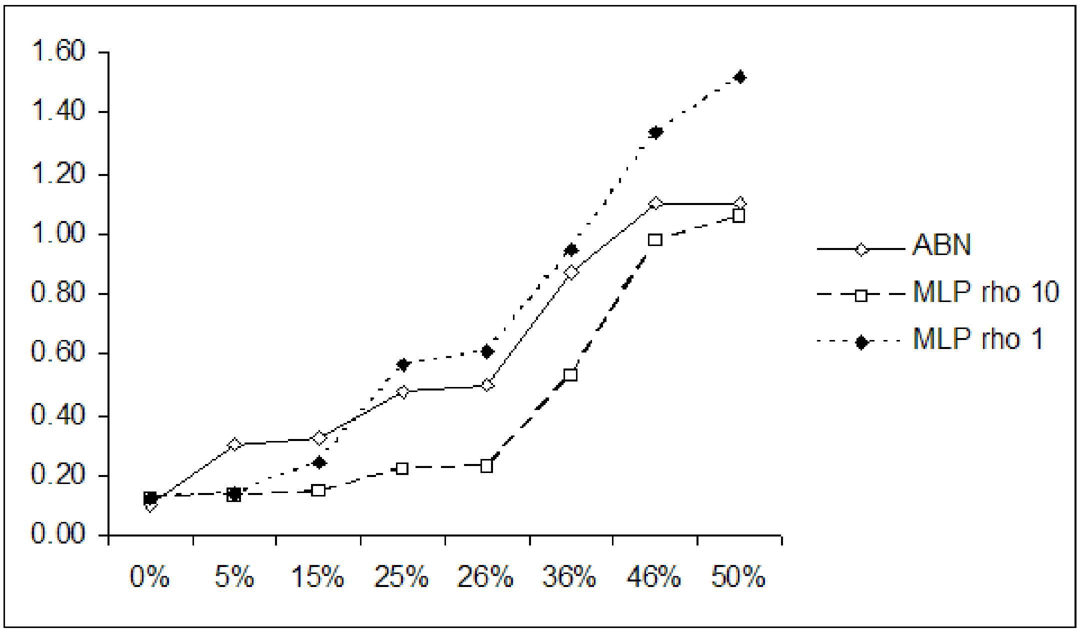
Similarly, both this network and the Pulse-Frequency Modulated Network were trained on the same task using a Genetic Algorithm (GA). The GA had a population size of 50, and a 1% mutation rate. The mutation size was the parameter size being mutated, using a uniformly distributed random number. Selection was by roulette and each gene in the string had a 50% chance of crossing over with its mate. Figure 10 shows the error profile of the training. A similar graph was obtained for the Pulse-Frequency Network.

Figure 10. Error Verses Epoch for a Genetic Algorithm trained Pulse-Width Modulated Network.



Finally, these results were compared to those generated by a MLP of exactly the same size. The comparison was done by adding noise to the patterns and observing the network response. This is shown in figure 11 (x axis – added noise, y axis – error).

Figure 11. Comparison of MLP and ABN (Pulse-Width Modulated).

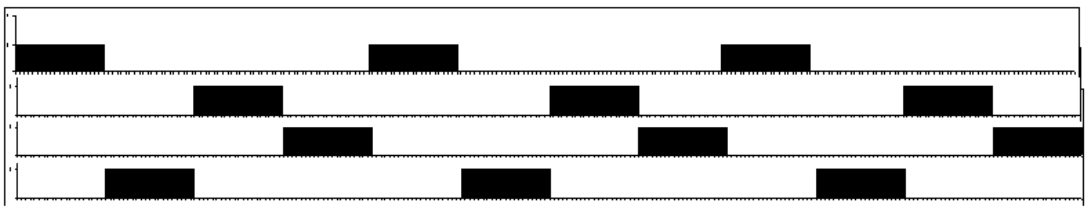


The slight differences between the networks can be accounted for in two ways. Firstly, because of their different (random) initial weights and secondly because the ABN signals are quantised into ten time periods - which limits their resolution (see figure 7).

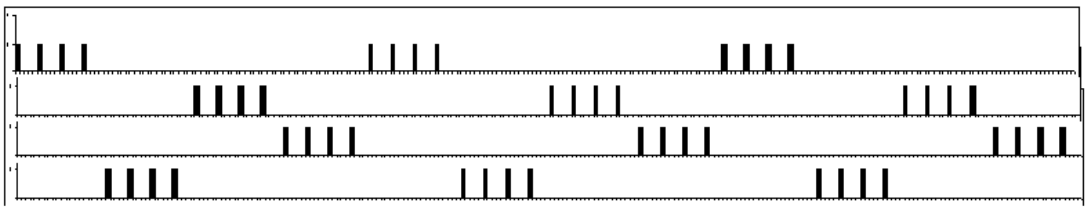
7.3 Waveform generation

As mentioned above, not only can the ABN networks provide Pattern Recognition, they can also generate waveforms. A good illustration of such a facility is the production of locomotive gait patterns for legged robots. Such patterns are used to drive the legs of the robot in the correct order to produce movement and are produced by networks termed Central Pattern Generators or CPGs. A number of ABNs were set up to fulfil this function and trained with simple user-selected GAs. The waveforms produced are self generated by the network - no external clock is required. Figure 12a shows a walking gait generated in this way using Pulse-Width Modulated Neurons arranged in a two-layer feedforward structure with 4 neurons in the first layer and 2 in the second. Figure 12b, is the same gait but generated in a network using both Pulse-Width and Pulse-Frequency Modulated units. This network required 4 further units and has the advantage that it controls not only the phase and stride length of the legs, but also their stride speed as well.

Figure 12a,b. Example of a Walking Gait.



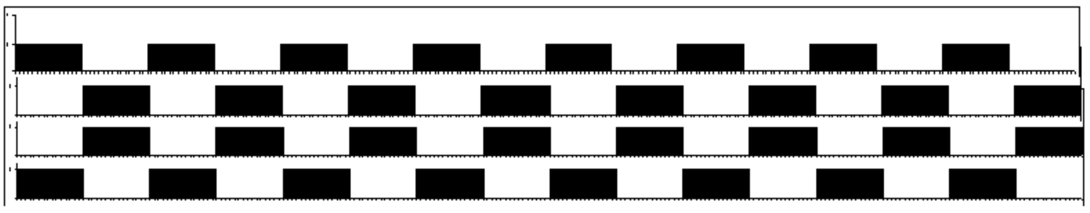
a) Pulse - Width neurons



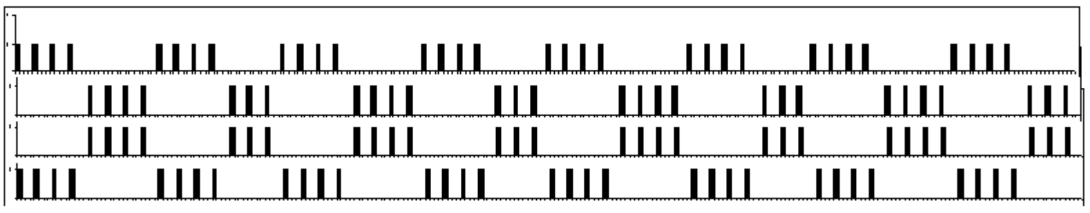
b) Pulse - Frequency neurons

All the common quadruped gaits were generated in this way including Pace, Gallop and Pronk. Figure 13a and 13b demonstrate a trotting gait as a further illustration. This network required 4 neurons

Figure 13a, b. Example of a Trotting Gait.



a) Pulse- Width neurons



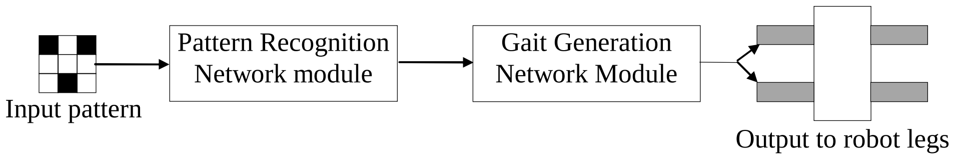
b) Pulse- Frequency neurons

One interesting observation made during these experiments was that, although such gaits can be produced by the ABN units alone, using the scheme outlined above, smaller networks generally result if simple AND and OR units are also allowed. This allows the pulse-width units to gate the pulse-frequency units and so produce the type of cyclical combination pulses shown above more easily.

7.4 Modular networks

As a final test of the ABNs flexibility, the outputs of the Pattern Recognition Network was used to trigger the CPG network. The idea of this was that different patterns could be programmed to correspond to different gaits. It also allowed the network to behave in a modular fashion (Muthurman, 2005; Macleod, 2009) and demonstrated the ability of the networks to behave as the "input" or "output" types mentioned above, using a single flexible unit type. This experiment worked well and four gaits could be triggered by the four different trained patterns. Figure 14 shows this structure.

Figure 14. A modular combination of the networks.


Input pattern
Pattern Recognition Network module
Gait Generation Network Module
Output to robot legs

8. Discussion and Conclusions

It should be apparent, from the examples shown above, that the ABN network is capable of exactly the same mappings and transformations (in the mathematical sense) as a similar Perceptron Network. Like the Perceptron, it may also be trained using an unmodified Back-Propagation Algorithm. In simple pattern recognition tasks there would seem to be little advantage, therefore, in using it. It requires the inputs to be converted into time-varying signals and, similarly, for the outputs to be read in this manner.

However, Pattern Recognition is only half the story. In many control and other engineering systems (for example, a DC motor controller or other, similar, actuators) the pulse-width or frequency modulated outputs of the network may be directly interfaced to the system. It is effectively operating in these engineering systems' "native language." The time varying nature of its operation also has similar advantages when dealing with time-varying input signals.

Of course, some other models, most obviously Spiking Neuron models, are also used to achieve similar ends. However, these models are based strongly on biological systems. They are therefore computationally and conceptually complex and difficult to program (Capanni, 2006) and also lack the flexibility required, in terms of producing flexible pulse-width and pulse-frequency signals, of useful engineering system (spiking models generally only produce inverse frequency modulated outputs).

These advantages are perhaps best illustrated in robot systems. This is a model which can be used for the sensory system (as a pattern recogniser), to generate the appropriate gait patterns for the robot's legs (as part of a CPG network) and finally to control the motors moving these legs. Such flexibility means that the designer does not have to choose different neuron models for different subsystems. The advantages of this are obvious and may also be seen when considering recent advances in modular evolution, where new modules build up automatically on older ones (Muthuraman, 2005; MacLeod, 2009), this work uses the models described above. Again, there are great advantages in having a neuron type which can handle complex mapping scenarios in such a situation.

References

Alberts, B et al (1994) Molecular Biology of the Cell (3rd ed), Garland Publishing Inc, New York, pp24 - 25, 111- 135, 721 - 782.

Capanni, N et al (2005) Artificial Biochemical Networks. In Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation - CIMCA'2005, Vienna, November 2005, IEEE, New York, Vol 2, pp98 - 102

Curtis, H (1969) The marvellous animals: an introduction to the Protozoa, Heinemann Education, London.

Capanni, N (2006) The Functionality of Spatial and Time-Domain Artificial Neural Models, PhD Thesis, The Robert Gordon University, Aberdeen.

Grestner, W and Werner, K (2002) Spiking Neuron Models, Cambridge University Press, Cambridge.

Gurney, K (1997) An introduction to neural networks, University College London Press, London. pp20-24.

Hopkin, K (1996) dot dot dot dash dash dash, New Scientist Magazine, vol 150, 2030. pp 40 – 43.

MacLeod, C et al (2009) Incremental Growth in Modular Neural Networks, Engineering Application of Artificial Intelligence, Elsevier, Amsterdam, Vol 22, 4-5, pp 600 – 666.

Muthurman, S (2005) The Evolution of Modular Artificial Neural Networks, PhD Thesis, The Robert Gordon University, Aberdeen.

O’Shea, M (2005) The Brain: A Very Short Introduction, Oxford University Press, Oxford. pp42 - 46.