Evolving a Minimally Constrained Image Recognition Neural Network

Aitkenhead, M.J.

The Macaulay Institute, Craigiebuckler, Aberdeen AB15 8QH, Scotland, UK, E-Mail: m.aitkenhead@macaulay.ac.uk

Keywords: *Image recognition; neural network; modularity; evolutionary computation.*

EXTENDED ABSTRACT

Artificial neural network-based applications have yet to live up to the promises made during their earlier theoretical period. There have been many small-scale applications, and the technology has been widely applied but as yet there have been no real autonomous systems demonstrated that are capable of truly sophisticated learning or complex behaviour. The reason for this apparent failure to deliver on earlier promises, and the reason therefore that artificial neural networks are no longer as fashionable as they once were, is postulated to be the constraints that are placed upon neural network models by their designers, with fixed architecture, dynamics and learning rules being applied in ways that prevent all but a small proportion of NNs' full potential to be realised.

Here, a model of neural network growth and activity is described that is intended to minimise the constraints of design and which uses evolutionary methods to develop a working system most suited to a specified task. The design of the network model follows broadly biologically plausible lines, with modularity, uptake and release of hormones, and self-detection of the model's actions allowing behavioural development through feedback. The method is still undergoing development, but has been demonstrated to work effectively when applied to a test case.

The constraint-minimisation methodology is applied to the problem of optical character recognition, and demonstrates the ability to develop rapid and accurate abilities in the area. The system developed is designed to react to a combination of simulated visual and audio inputs, responding to these inputs through audio output that is used to develop a feedback system. Through this method, the evolved network becomes capable of mimicking the audio inputs and following a period of training, of responding correctly to the visual inputs without audio prompting.

It is seen as unavoidable that some design specification, and therefore constraint, is necessary in order to develop the network. The problem lies in identifying the network components and design parameters that are necessary, and the relationships between these different aspects of the system. Once this has been achieved in such a way that the design of the system is as flexible as possible, it is then evolved from an initial design using a definition of fitness that corresponds to an ability to mimic audio inputs and respond to visual outputs correctly. The components and parameters that have been identified as necessary to the system include the following:

- Nodes (parameters include location, internal chemical levels, input activation and output activation)
- Synapses (parameters include input and output node identification, connection weight, type and internal chemical levels)
- Architecture (this includes considerations of modularity, connectivity, module spatial dimensions and module population size)
- Node development (factors include node growth and node removal)
- Synapse development (factors include synapse growth and synapse removal)
- Hormone density (which is affected by diffusion rates, density differentials, node and synapse hormone production)

The application developed here does not yet use all of these considerations (specifically, node and synapse growth and removal are not implemented). The evolutionary development of the network takes place over four epochs of successively more complex inputs and output requirements. The ultimate aim of the system is to provide a base for further development of semi-autonomous neural network learning systems capable of interaction with the real world.

1. INTRODUCTION

The design of a successful neural network evolution system is elusive, with systems developed for solving specific problems, but nothing yet capable of providing a general solution. Nolfi & Parisi (2002) discuss the wide variety of methods that can be employed in evolving artificial neural networks, ranging from mutation of the individual connection weights to evolution of the architecture and learning strategy.

In particular, any system capable of providing a network that can generalise will have to avoid constraining the design to fit the problem. Therefore, researchers need to identify each of the parameters that are of real importance within a neural network, and then identify which other parameters can affect them. The relationships could then be expressed in a manner that allows them to be evolved, and the network could be initialised as a modular system and which develops into a network that can handle the inputs given to it. Dinerstein *et al.* (2003) emphasise the need to have modularity in a neural network capable of performing complex tasks.

Optical character recognition (OCR) is a common goal of artificial intelligence, with the ability to read handwritten documents or identify characters within a real situation, such as car number-plates, being extremely useful. Avi-Itzhak (1995), Guyon (1991), Gatos *et al.* (1993) and LeCun *et al.* (1990) amongst others, applied neural networks to optical character recognition. The development of OCR capabilities is used here as a target to demonstrate the capabilities of the neural network method used. In addition mimicry, a behaviour commonly seen amongst animals, will be another of the system's goals.

The training method that is used must be as flexible as possible, allowing many different types of learning. In order to develop a system capable of avoiding or at least minimising design constraints, the degree of flexibility inherent within the system must be sufficient to encompass as broad a range of designs as possible. A comparison of different neural network algorithms applied to the problem of OCR (Van der Smagt, 1990) showed that learning and accuracy rates were sensitive to changes in network structure and training technique. However, it is impossible to avoid specifying any aspects of the design, so therefore the specification must be very general. One modelling method that has been recognised as providing the ability to implement a very wide range of mathematical functions is feedforward neural networks. In this work, the interactions between parameters is modelled using neural networks that are evolved to provide a system capable of learning. Parameters, the presence or absence of relationships between these parameters and the location of input and output components are the only design specifications given to the system. The rest is kept flexible.

García-Pedrajas *et al.* (2001) demonstrated a novel evolutionary approach for neural network development that used crossover between genotypes on a radial basis function network. Various difficulties remain to be solved with the method of evolving designs, rather than the implementation of evolutionary pressures on the genotypes themselves (Stanley & Miikkulainen, 2002). In terms of the imagery supplied to the network, this is kept relatively simple, with character images corrected to align them within the field of view. A combination of image correction and neural network methods was shown to improve classification (Chowdhury *et al.*, 2002).

2. METHODS

Several considerations must be used when designing a neural network system through evolution. These can usually be broadly categorised into component design, component dynamics, network architecture, evolutionary methods and network training algorithms.

2.1. Network component design

Although the network design is required to be as general as possible, certain design specifications must be given in order to start from somewhere. The following system components, parameters and functionalities have been identified as necessary and integral to the design of the system, with each component being affected by a range of others:

Nodes

- Location – fixed
- Internal factors – affected by node internal factors, input activation, output activation, local hormone density
- Input activation – affected by connecting synapse activation
- Output activation – affected by input activation, node internal factors, local hormone density

Synapses

- Input node - constant
- Output node - constant

- Weight – affected by weight, type, internal factors
- Type – affected by type, internal factors
- Internal factors – affected by type, weight, signal sent, internal factors

Initial design
- Modularity - initialisation
- Connectivity - initialisation
- Module size – initialisation
- Module population - initialisation

Node development
- Node growth – affected by local hormone density, local node density
- Node removal – affected by local hormone density, local node density, local synapse density, internal factors

Synapse development
- Synapse growth – affected by local hormone density, local node density, local synapse density
- Synapse removal – affected by local hormone density, local node density, local synapse density, internal factors

Hormone density
- Density – affected by neighbourhood density differential, diffusion rate, local node activations, type

There is an obvious desire for the system to be as flexible as possible. Each factor is numbered and identified, and has a value and a list of factors identified that can affect it. When a new factor is created, for example using node growth, a subroutine creates a corresponding list, and when a factor is removed another subroutine removes it. A specific set of factors forms the input node activations, and another forms the output node activations. For each cell in the spatial array which contains the network, there is a count of the number of nodes and the number of synapses in that cell, and this count is adjusted to deal with additional nodes and synapses. For each node there are total of ten factors to be considered, and for each synapse there are eleven factors.

Development of the network therefore requires the use of a large list of factors and an equally large list of network weightings. Training of the network takes place, as described below, through the activation of the input node factors, which is then followed by the adjustment of every single factor in the network. For a network of over 200 nodes and over 2000 synapses which occurs during the final stages of evolution, this means a total of >26000 factors are calculated using the secondary neural networks, requiring approximately 2 million calculations. It is therefore necessary to keep these calculations as simple as possible to cut down on processing time, by avoiding complex secondary neural network node activation calculations.

2.2. Secondary networks

For each of the above parameters identified as a required network component, there are initialisation settings, and a secondary neural network which provides the relationships between the factors and all factors that can affect them. This secondary neural network is relatively simple and small, and subject to evolution in the same way that the initialisation settings are. The number of hormone types is set at three, and the number of synapse types at three. There are also three node internal factors, and three synapse internal factors for each type. This means a total of twenty networks each with a number of inputs ranging from 1 (node input activation) to 8 (several factors). Each secondary network has one hidden layer, each with ten nodes. So there are approximately 1000 values describing the dynamics of the system, with perhaps 200 additional values for the initialisation. While this is a large number of factors, it is well within the limits of evolutionary methodologies as far as optimisation is concerned, and is relatively small compared to systems where the weights of connections are themselves evolved.

2.3. Network modularity

The internal structure of the network is known to have a strong impact on its performance and behaviour. Modularity, or the splitting of the network into several sub-compartments, has been shown effective as a method of generating lifelike behaviour, with hierarchical patterns of response in which different individual actions can be combined in many ways. The network design here is modular, with specific sections of the network corresponding to specific actions being defined in terms of size and location, and with other modules being subject to alteration by the evolutionary process. The input module corresponding to vision occupies the y=0 surface or ‘wall’ of a cube of side length 1, with the audio inputs being situated in the centre of the z=1 surface and the audio outputs in the centre of the z=0 surface. The remaining modules are left to occupy the internal space of this cube, as shown in Figure 1.

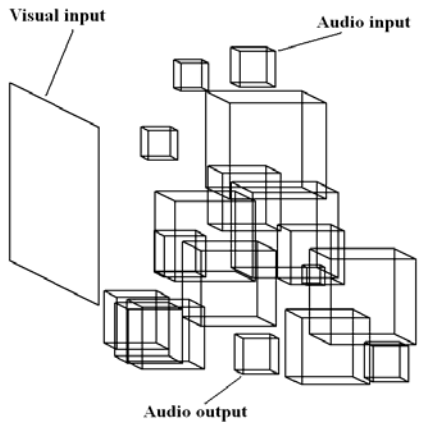

Figure 1. Modularity design of the neural network

Various parameters within the neural network relate directly to the modularity of the system. These include the population of each module and its dimensions (which are fixed for the input and output modules), and the connectivity both between nodes within a module, and between nodes in different modules. The population of each module is limited using maximum and minimum values, and the dimensions are restricted to the range [0, 1]. Connectivity between nodes, both within a module and between different modules, is defined using two parameters. The first (X) is a probability of connection for two nodes if there is no distance connecting them, and the second (Y) is a value giving an exponential rate of decay of this probability with distance. Equation 1 gives the relationship defining the probability P of connection of two nodes using these parameters.

$$P = Xe^{-Y} \qquad \text{(eqn. 1)}$$

2.4. Evolutionary methods

A standard evolutionary methodology is applied, with the network's level of fitness being measured and used to determine whether the mutations are accepted. There is a small annealing allowance to provide the network with the ability to jump out of local minima, corresponding to the simulation of temperature within an annealing substance. Following measurement of the network's fitness, this fitness value is compared to the best value obtained so far. If the current fitness value is better than the best so far, or is within the annealing allowance, then the latest mutations are kept. If not, then the mutations are discarded.

2.5. Network training & design

During each evolutionary phase, there is a single network being trained at any one time. The training takes the form of a series of a set number of time slices (1000), with each set having a fixed length and being called a 'step'. In each step there are 10 time 'quanta' during which a single activation-adjustment cycle. Each step will involve the network being given one input set, and being trained to give the appropriate output. This will mean that each network will have a total of approximately 10000 training quanta, with each quantum involving the activation of every neuron and synapse in the network. It is estimated that each synapse will require 1000 Hz of processing time, and that the maximum number of synapses will be approximately 2000. This means that the number of processing cycles required for each generation will be between $2x10^{10}$ processing cycles, which on a high-end desktop PC takes approximately 20 seconds. The problem here lies in balancing the number of evolutionary steps with the size of the network in order to optimise the system's development. If the network did not have to be evolved, then it would be possible to get away with having as many as 100000 synapses running at full speed, which would allow approximately 10000 nodes and therefore more sophisticated behaviour.

Each training step includes the random selection of one of the character images, and the transmission of this information to the network throughout the step. Additionally, the input to be mimicked is given for the first five quanta, and looked for during the second five (giving a total of ten time quanta for each step). The controlling algorithm will only look for the output during the second five-quanta segment, and will ignore outputs given during the first five. For the entire evolutionary algorith, the following program structure is used:

```
Initialise arrays
Initialise variables
Initialise pre-evolutionary settings
Loop through generations
   Initialise network
   Loop through training steps
      Give input
      Determine node settings
      Determine synapse settings
      Adjust node population
      Adjust synapse population
   Measure network fitness
   Subject network to evolutionary pressures
   Display results
   Save results
```

2.6. Input data

The ICDAR 2005 Robust Reading website (http://algoval.essex.ac.uk:8080/icdar2005/index.jsp) is

the site from which the datasets were obtained. For each number, a total of 50 images were selected to train the neural network. All that is available from the website at the moment is numbers. The first step involved translating all of the images into ascii files. This has been done, with each image now occupying a text file which has 8x8 values in greyscale.

Audio input corresponding to each character was obtained by recording a male voice saying the numbers 0 to 9. The recordings were translated into text files with eight columns of values over five time steps, with each column corresponding to a frequency range (0-50 Hz, 50-100 Hz, 100-200 Hz, 200-500 Hz, 500-1000 Hz, 1000-2000 Hz, 2000-5000 Hz, 5000-10000 Hz). The values in each cell were obtained by integrating the Fourier transform of the audio file, sampled at a rate of 22050 Hz, over each range and adjusting to fit on the range [0, 1].

2.7. Developmental epochs

The network development took place over several epochs, each of which comprised successively more complex training data. During the first epoch, only two inputs were given to the network, with one of the inputs activated and the other inactive. Successive training sets were used throughout each generation to train the network to mimic the single input that was being given. This developmental epoch was continued, with the network evolving to the point where it could successfully repeat the given input (i.e. by the end of the training session, if input node 1 was active and input node 2 was inactive, then the network would give an output of the same kind).

Following the first developmental epoch, the network was evolved further using an additional three, each more complex than the last. Epoch two involved the use of four inputs, with four possible input node variations (1100, 1010, 0101, 0011). Epoch three involved the use of twenty possible inputs, with ten randomly selected input sets each with four 1s and five 0s. The final epoch involved the use of the 10 optical character input sets, each of which comprised 64 input variables given over five time steps.

3. RESULTS

Initial results during the first epoch showed that the network was evolving, but that it was not doing so in a manner that led to consistently better performance than a random initialisation. The range of fitness values increased greatly from the initial settings, but were equally likely to be lower than the original fitness value than higher. This was discovered to be due to the overall system design, in which auditory outputs were fed back into the auditory input module in order to interfere with the externally supplied auditory input. The rationale behind this was that if the network's audio output matched the external audio input, then there would be a reinforcement of input which would lead to the network being more likely to produce the correct output. If the auditory output did not match with the external audio input, then the interference would prevent attractors forming in the neural network. However, this was found to be the case only if the audio input and output node activations were adjusted to provide a constant total activation (i.e. total output was constant).

Following adjustment to the network design, the first epoch using only two inputs evolved to a fitness level of 75% in 4523 generations, at which point the second epoch was initiated. Figure 2 shows the fitness level development throughout the first epoch.

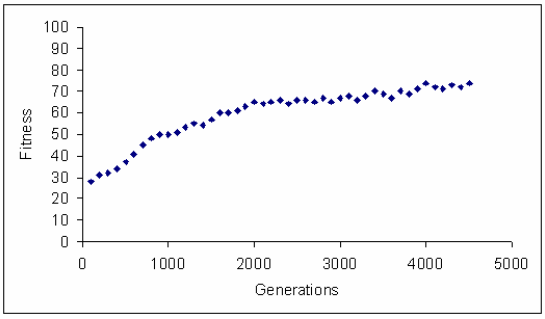

Figure 2. Fitness evolution during the first evolutionary epoch

The second and third epochs took 1894 and 7747 generations respectively to reach the 75% fitness threshold required for epoch succession. Epoch four was implemented for a total of 10000 generations, by the end of which the greatest fitness level reached was 54%. This level of fitness appeared to be a plateau, as can be seen in Figure 3.

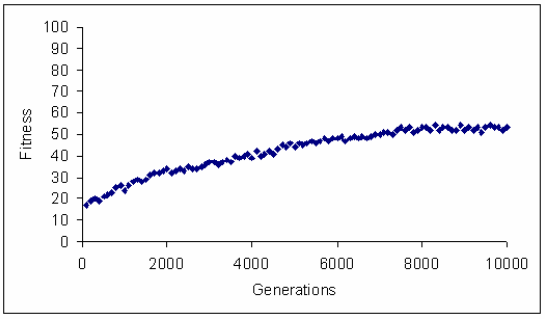

Figure 3. Fitness evolution during the fourth evolutionary epoch

For the network trained during the very last generation of epoch four, detailed examination of the accuracy levels and the ability to recognise specific characters was carried out. Accuracy was defined using a measure of the Hamming distance of the actual output from the audio input given. The Hamming distance was measured to each of the possible audio inputs, and a measure of how often the audio output given was closest to the target output. It was found that the network was capable of achieving the correct response using these criteria for 6 out of the 10 characters (1, 2, 4, 5, 7, 8), with the Hamming distances to the target output taking 2nd place for the characters 3 and 6, 3rd place for the character 0 and with the worst position being 5th place, for the character 9.

4. DISCUSSION

Development of a method of evolving neural networks using a relatively constraint-free design has been shown possible in this work. The developed network is capable of learning to mimic simulated audio inputs from a virtual trainer, and can recognise optical characters while providing an output corresponding to the given audio inputs. The extremely flexible design of this system means that it could be applied to a wide range of situation where neural network training is relevant, and provides the potential for more sophisticated NN development than was carried out here.

Initial attempts to develop the network did not prove as effective as intended, with adjustments being required in order to improve both the evolutionary algorithm and the input/output feedback concept. These corrections to the system show that there is still a requirement to consider the design that is being used, with the original concept constraining the capabilities of the network to evolve and satisfy the fitness requirements placed upon it.

5. REFERENCES

Avi-Itzhak, H.I., Diep, T.A., Garland, H. (1995), High accuracy optical character recognition using neural networks with centroid dithering. *IEEE Transactions on Pattern Analysis and Machine Intelligence Volume 17 , Issue 2 (February 1995)*, pages 218–224.

Chowdhury, A.A., Ahmed, E., Ahmed, S., Hossain, S., Rahman, C.M. (2002), Optical character recognition of Bangla characters using neural network: a better approach. *2nd International Conference on Electrical Engineering (ICEE 2002)*, Khulna, Bangladesh.

Dinerstein, J., Dinerstein, N., de Garis, H. (2003), Automatic multi-module neural network evolution in an artificial brain. *NASA/DoD Conf. on Evolvable Hardware, Chicago, Illinois, USA, July 9-11, 2003.*

García-Pedrajas, N., Sanz-Tapia, E., Ortiz-Boyer, D., Hervás-Martínez, C. (2001), Introducing Multi-objective Optimization in Cooperative Coevolution of Neural Networks. *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks: Connectionist Models of Neurons, Learning Processes and Artificial Intelligence-Part I Pages: 645 – 652.*

Gatos, B., Karras, D., Perantonis, S. (1993), Optical character recognition using novel feature extraction & neural network classification techniques. *In P.J.G. Lisboa and M. J. Taylor (eds.) Proceedings of the Workshop on Neural Network Applications and Tools (Liverpool, UK, 13-14 September 1993), 65-72. IEEE Computer Society Press.*

Guyon, I. (1991), Applications of neural networks to character recognition. *International Journal of Pattern Recognition and Artificial Intelligence 5, 353-382, 1991.*

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D. (1990), Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems 2* (D.S. Touretzky, ed.), pp. 598-605, San Mateo, CA, Morgan Kaufmann, 1990.

Nolfi, S, Parisi, D. (2002), Evolution of artificial neural networks. In *Handbook of brain theory and neural networks, MIT Press, pp. 418--421.*

Stanley, K.O., Miikkulainen, R. (2002), Efficient evolution of neural network topologies, *Proceedings of the 2002 Congress on Evolutionary Computation* (CEC '02). Piscataway, NJ: IEEE, 2002.

Van der Smagt, P. (1990), A comparative study of neural network algorithms applied to optical character recognition. *Proceedings of the third international conference on Industrial and engineering applications of artificial intelligence and expert systems - Volume 2. Charleston, South Carolina, United States Pages: 1037 – 1044.*