THE EVOLUTION OF GATED SUB-NETWORKS AND MODULARITY IN THE HUMAN BRAIN

JOHN A. BULLINARIA
School of Computer Science
The University of Birmingham
Birmingham, B15 2TT, UK
E-mail: j.bullinaria@physics.org

Few people would disagree that the human brain is modular, but there is less agreement on the reasons why it has evolved to be like that. Recently I re-examined the Rueckl, Cave & Kosslyn study[9] which demonstrated the advantages of having a modular architecture in neural network models of a simplified version of the "what" and "where" vision tasks. Explicit evolutionary simulations confirmed that the advantage can cause modularity to evolve, but also demonstrated that simply changing the learning cost function produced a system that learnt even better than before, and in which modularity did not evolve. In this paper I attempt to find a more robust characterisation of the evolution of modularity in terms of gated sub-networks (i.e. mixtures of expert networks). Once again, a careful analysis of a systematic series of explicit evolutionary simulations indicates that drawing reliable conclusions in this area is not as straightforward as it might at first appear.

1 Introduction

The human brain is undoubtedly modular[10], and there are numerous reasons why it might have evolved to be that way. Given the likelihood of disruptive interference if two tasks are carried out by a single system, rather than by two dedicated modules, the evolution of modularity might seem inevitable. Indeed, in a well known paper, Rueckl, Cave & Kosslyn[9] demonstrated explicitly the advantages of having a modular architecture (i.e. specialised sub-sets of hidden units) in neural network models of a simplified version of the "what" and "where" vision tasks. There is good corresponding neuroscientific evidence of distinct cortical pathways in the human brain for these two tasks[7,4]. The implication is that the computational advantages would cause any process of evolution by natural selection to result in such a modular architecture, and hence answer the question of why modularity has arisen. In a recent study[3], I carried out the explicit simulations of that evolutionary process. As expected, I confirmed that the advantage can indeed cause modularity to evolve. However, I also demonstrated how simply changing the learning cost function produced a system that learned even better than before, and in which modularity did not evolve. In another paper on the same simplified "what" and "where" vision tasks, Jacobs, Jordan & Barto[6] showed how a

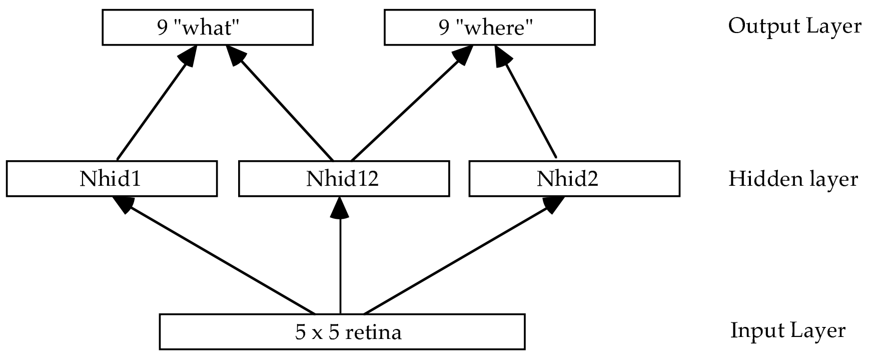

Figure 1. Architecture of the basic network model for the "what" and "where" tasks.

network of gated expert sub-networks would *learn* how to perform the two tasks in a modular manner. In this paper I attempt to find a more robust characterisation of the *evolution* of modularity in terms of these gated sub-networks (i.e. mixtures of expert networks). Once again, a careful analysis of a systematic series of explicit evolutionary simulations indicates that drawing reliable conclusions in this area is not as straightforward as it might at first appear.

2 The Neural Network Models

The Rueckl et al. study[9] was based on a standard three layer feed-forward network with sigmoidal activation functions and on-line gradient descent learning. It mapped a simplified retinal image (a 5×5 binary matrix) to a simplified representation of "what" (a 9 bit binary vector with one bit "on") and a simplified representation of "where" (another 9 bit binary vector with one bit "on"). Each of the 9 training images consisted of a different set of 5 cells "on" within a 3×3 sub-retina array, and the 9 positions corresponded to the possible centres of a 3×3 array within the full 5×5 array. To enable direct comparisons with the previous studies, I used exactly the same 9 input patterns and 9 positions, giving the same 81 retinal inputs for training on, in my corresponding evolutionary simulations[3]. Figure 1 shows the basic network with arrowed lines representing full connectivity, and $Nhid1$, $Nhid12$, $Nhid2$ specifying how many hidden units in each block. Rueckl et al.[9] compared the fully distributed network ($Nhid1 = Nhid2 = 0$) with the purely modular network ($Nhid12 = 0$). If the maximum number of hidden units $Nhid = Nhid1 + Nhid12 + Nhid2$ is fixed, varying the two parame-

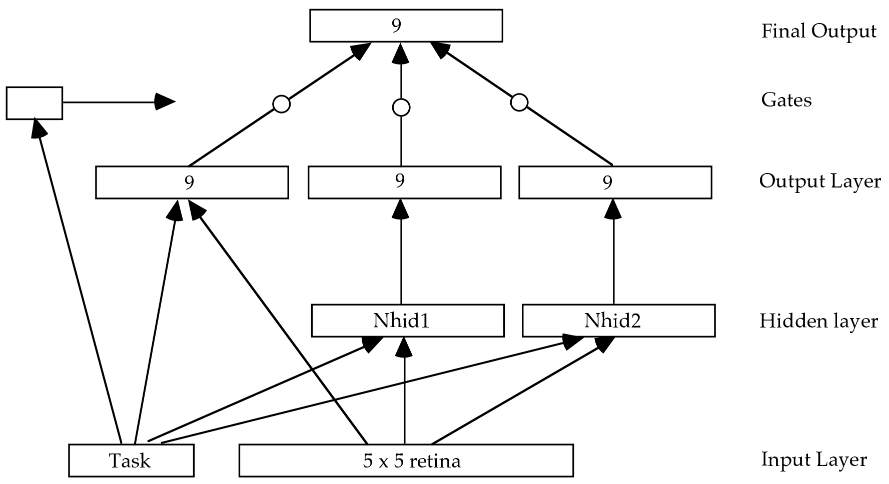

Figure 2. Architecture of the gated network model for the "what" and "where" tasks.

ters $Nhid1 + Nhid12$ and $Nhid2 + Nhid12$ enables an exploration of the full continuum between these extremes.

Jacobs et al.[6] employed the slightly more complex architecture illustrated in Figure 2. Here we have an extra input unit specifying the task and just one final set of 9 output units, and we train on 162 examples (81 for each task). The system is comprised of three separate fully connected "expert networks" of the standard form (with 0, $Nhid1$ and $Nhid2$ hidden units respectively), each producing either a 9 bit "what" or "where" output depending on the setting of the task input unit. The novel feature of this architecture is that the task unit also drives a series of "gating parameters" g_i that specify the extent to which the output layer of constituent network i contributes to the final output for the given task. If the output vectors of the three sub-networks are $\mathbf{y_0}$, $\mathbf{y_1}$, $\mathbf{y_2}$ then the final output is $\mathbf{y} = g_0\mathbf{y_0} + g_1\mathbf{y_1} + g_2\mathbf{y_2}$. For each task we constrain $\sum_{i=0}^{2} g_i = 1$, and so modularity corresponds to having one $g_i = 1$ and the rest zero for one task, and a different $g_j = 1$ and the rest zero for the other task. If we have the same $g_i = 1$, and the rest zero, for both tasks, we have a non-modular architecture with one constituent network doing everything. If the g_i's do not become binary we have a range of modular and non-modular possibilities. In the Jacobs et al. study[6] they set $Nhid1 = 18$ and $Nhid2 = 36$ and used an on-line gradient descent algorithm to simultaneously train the weights and gates. Their simulations showed how the system learns to use a modular architecture, with the no-hidden layer network handling the linearly separable "where" task, and the 36 hidden unit network handling the harder "what" task.

3 Learning and Modularity

While a network is learning, a hidden unit that is processing information for more than one output unit is likely to receive conflicting weight update contributions for the weights feeding into it, with a consequent degradation of performance relative to a network that has a separate set of hidden units for each output unit[8]. However, employing the extreme version of modularity that has a dedicated set of hidden units (or module) for each output unit is likely to be rather inefficient in terms of computational resources, and an efficient learning algorithm should be able to deal appropriately with the conflicting weight update signals anyway. Nevertheless, splitting the hidden units up into disjoint sets corresponding to distinct output tasks, may be an efficient option.

This compromise certainly appears to be appropriate for the networks described above. Rueckl et al.[9] demonstrated how the modular version of the network in Figure 1 performed better than the fully distributed version. Then Jacobs et al.[6] showed how the gated network of Figure 2 could *learn* to process two the tasks in a modular fashion with advantages in terms of learning speed, minimizing cross-talk (i.e. spatial interference), minimizing forgetting (i.e. temporal interference), and generalisation. In this paper, I shall explore the idea that such computational advantages are sufficient to drive the *evolution* of gates that correspond to modularity. The assumptions that the two tasks are easily recognised, and that the appropriate gates are easily operated, will be implicit throughout.

4 Simulating Evolution

Simulating an evolutionary process for the neural network models discussed above is simply a matter of taking a whole population of individual instantiations of each model and allowing them to learn, procreate and die in a manner approximating these processes in real (living) systems. I take a more natural approach to procreation, mutation and survival than many evolutionary simulations have done in the past[1]. Instead of training each member of the whole population for a fixed time and then picking the fittest to breed and form the next generation, I allow the populations to contain competing learning individuals of all ages, each with the potential for dying or procreation at each stage. During each simulated year, each individual will learn from their own experience with the environment (i.e. set of training/testing data) and have their fitness determined. Assuming all other factors are equal, the evolutionary fitness can be taken to be a simple function of the task fitness used by the learning algorithm. A biased random subset of the least fit individuals, together with a flat random subset of the oldest individuals, will then die. These are replaced by children, each having one parent chosen randomly from the fittest members of the population, who randomly chooses a mate from the rest of the whole population. Each child inherits characteristics from both parents such that each innate free parameter is chosen at random somewhere between the values of its parents, with sufficient noise (or mutation) that there is a reasonable possibility of the parameter falling outside the range spanned by the parents. These simulations could undoubtedly be made more realistic, but for the purposes of the current study, this simplified approach seems adequate. A similar regime has already been employed successfully elsewhere to study the Baldwin effect in the evolution of adaptable control systems[2].

5 Simulation Results

The simulated genotype of each individual will include all the innate parameters needed to specify the network details, namely the architecture, the learning algorithm, the learning rates, the initial connection weights, and so on. In real biological evolution, all these parameters will be free to evolve. However, for the purposes of simulation efficiency, it generally makes sense to be more restrictive. Here it is appropriate to allow the architecture to evolve, but have each individual start with random initial connection weights. Since the optimal learning rates will vary with the architecture, we must allow these to evolve along with the architecture.

It is clearly important to fix the evolutionary parameters appropriately according to the details of the problem and the speed and coarseness of the simulations. For example, if all individuals learn the task perfectly by the end of their first year, and we only test their performance once per year, then the advantage of those that learn in two months over those that take ten is lost and our simulated evolution will not be very realistic. Since the networks were allowed to evolve their own learning rates, this had to be controlled by restricting the number of training data presentations per year to 10 for each individual. Choosing a fixed population size of 200 was a trade-off between maintaining genetic diversity and running the simulations reasonably quickly. The death rates were set in order to produce reasonable age distributions. This meant about 5 deaths per year due to competition, and another 5 individuals over the age of 30 dying each year due to old age. The mutation

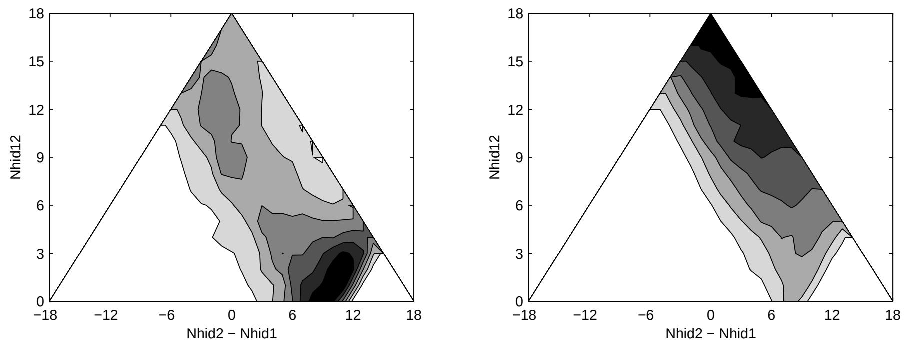

Figure 3. Mean Learning Times for the Basic Networks: (a) SSE, (b) CE.

parameters were chosen to speed the evolution as much as possible by maintaining genetic diversity without introducing too much noise into the process. These parameter choices led to coarser simulations than one would like, but otherwise the simulations would still be running.

5.1 Basic Model

I have already presented the results of the simulated evolution of the basic model of Figure 1 in detail elsewhere[3]. However, to help interpret the results for the gated networks, it is worth summarising here what happens. The network and learning regime are the same as that used by Rueckl et al.[9] The total number of hidden units is fixed at 18, and we allow the learning parameters and the two architecture parameters $Nhid1 + Nhid12$ and $Nhid2 + Nhid12$ to evolve as described above. We find that all the innate parameters soon settle down to appropriate values. In particular, we end up with $Nhid12 \simeq 0$ which corresponds to a modular architecture. This is exactly what we would expect given the advantages that Rueckl et al. found for modularity. The interesting discoveries arose in the process of testing the robustness of this result. It was natural to try changing the gradient descent cost function from Sum Squared Error (SSE), as used by Rueckl et al.[9], to Cross Entropy (CE), which is generally more appropriate if the outputs are to be interpreted as probabilities[5]. Doing this resulted in a non-modular architecture evolving with $Nhid1 \simeq Nhid2 \simeq 0$, and the individuals learned the tasks significantly more quickly than before. Moreover, even within the SSE case, it was found that the evolution of modularity could be rendered less certain simply by increasing the total number of hidden units available[3].

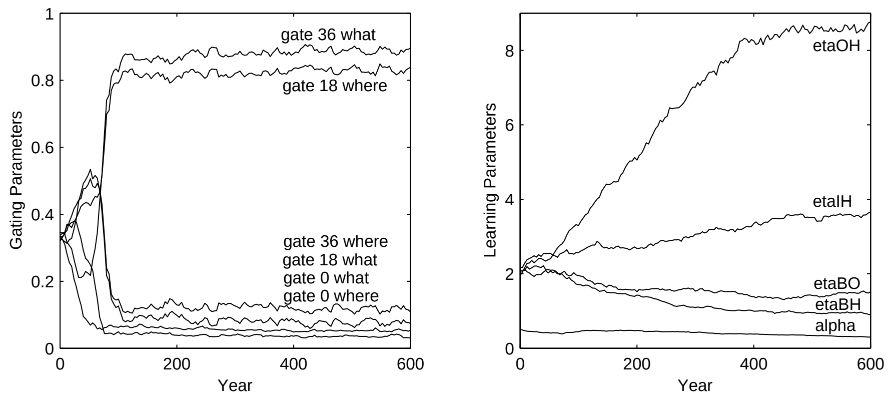

Figure 4. Evolution of the innate parameters for the Gated Networks.

Further investigation revealed that we could come to a good understanding of what evolves simply by plotting the mean learning times as a function of architecture. Figure 3 shows contour plots for the SSE and CE cases with 18 hidden units in total. The fastest learners, shown darkest, are found for the same architectures that emerged from the corresponding evolutionary simulations.

5.2 Gated Model

The evolution of the gated networks is naturally more complex than that of the basic model. In this case the numbers of hidden units are fixed and the evolvable architecture parameters are the gates g_i. For each of the two tasks, there are three gates g_i constrained to satisfy $\sum_{i=0}^{2} g_i = 1$. Figure 4a shows how the population mean gates evolve. We see that the expected modularity emerges with the 36 hidden unit sub-network performing the "what" task and the 18 hidden unit sub-network performing the easier "where" task. All the other gates are very near zero. The learning parameters evolve as appropriate for their tasks. For example, Figure 4b shows how the learning rates and momentum evolve for the 36 hidden unit sub-network. Note that the four learning rates take on very different values - those for the biases ($etaBH$, $etaBO$) are lowest, that for the input to hidden layer ($etaIH$) is several times higher, and that for the hidden to output layer ($etaHO$) is several times higher again. If we had attempted to set these by hand, or forced them to all take the same value, we may not have allowed each sub-network perform at its

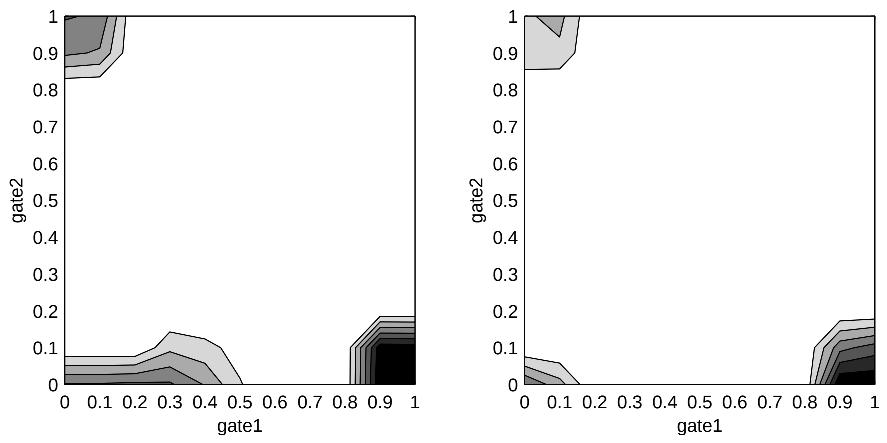

Figure 5. Mean Learning Times for the Gated Networks: (a) SSE, (b) CE.

maximum efficiency, and the emergent architecture may have been biased.

Unfortunately, with three constituent networks, there are too many free parameters to allow the plotting of the mean learning times as a function of architecture as we did in Figure 3. However, such plots are possible if we restrict ourselves to using just two of the three constituent networks. Naturally, it makes sense to use the two networks that emerged as the most useful in the evolutionary simulation, namely the sub-networks with 18 and 36 hidden units. Since the gates for each task must sum to one, we now only have two independent gate parameters which we can conveniently define to be $gate1 = g_{18}(where) = 1 - g_{36}(where)$ and $gate2 = g_{18}(what) = 1 - g_{36}(what)$. Now averaging over many training runs for each gate combination we find the mean learning time plots shown in Figure 5. Note that, in this case, using CE rather than SSE as the learning cost function makes little difference. The fastest learning times are seen at the bottom right corner of these plots, i.e. for $g_{18}(where) \simeq 1$ and $g_{36}(what) \simeq 1$, which is what we would expect given the results of the full evolutionary simulations. Note the presence of local minima in the top left corner, which corresponds to the modular architecture with the roles of the two networks reversed, and in the bottom left corner where the larger network takes on both tasks and we have a non-modular architecture. It is interesting to observe that if we do not include cross-over mutations in the the full simulated evolutionary process, that can occasionally swap the values of pairs of gates, the population can easily get stuck in one of these sub-optimal minima. Learning the gates by a gradient descent algorithm also tends to result in these local minima.

5.3 Pre-Sigmoidal Gated Networks

One could argue that given the way the network is set up, it is inevitable that binary gates will evolve. If the activation of unit j in the ith constituent network is a_{ij} and the connection weights into output unit k are w_{ikj}, the final network outputs will be

$$y_k = \sum_i g_i \text{Sigmoid}(\sum_j w_{ikj} a_{ij}).$$

Then, since the target outputs are binary, and the actual outputs take the form of a weighted average of the $\text{Sigmoid}(\sum_j w_{ikj} a_{ij}) \in [0, 1]$, the final output for a given task can never be better than that of the best sub-network. Consequently, given the constraint on the sum of the gates, there will always be a tendency for the gates g_i corresponding to the least accurate networks to decrease, and that of the most accurate network to increase. Once a gate starts to decrease, the corresponding sub-network gets a smaller contribution of the error signal and learns less well, so the gate is forced smaller still. Eventually the gates end up binary, and the modularity then follows from a simple consideration of best use of resources.

As always, it is appropriate to test the robustness of our results. Suppose instead that the sigmoid occurred at the final output, rather than at the outputs of the constituent networks, i.e.

$$y_k = \text{Sigmoid}(\sum_i g_i(\sum_j w_{ikj} a_{ij})) = \text{Sigmoid}(\sum_i \sum_j (g_i w_{ikj}) a_{ij}).$$

This is now mathematically equivalent to having the gates act as a task dependent modulation of the strengths of the final layer of connection weights. There is now nothing to force the gates to go binary and the network has a less restricted choice of best architecture. Indeed, if all the gates are equal, the sub-networks merge into one big homogeneous network, and there are values of the gates that render the gated network equivalent to the non-modular optimal architecture of our basic Rueckl et al. model. It should not be surprising then, that evolving this modified system leads to the emergence of somewhat different architectures to the previous simulations.

Figure 6 shows how the mean learning times depend on the architecture in this "pre-sigmoidal gate" case. The most obvious thing to notice is that

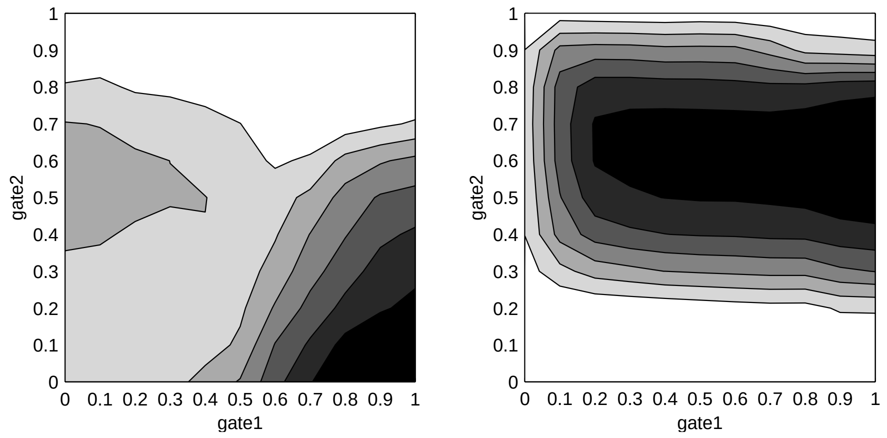

Figure 6. Learning Times for Pre-Sigmoidal Gated Networks: (a) SSE, (b) CE.

the difference arising from using CE rather than SSE has returned. In the SSE case, the fastest learners are in the same modular regime as before, and this emerges in the full evolutionary simulations as well. However, for CE we find the optimal gates in the region of $g_{18}(where) \simeq 1$, $g_{36}(where) \simeq 0$, $g_{18}(what) \simeq 0.6$, $g_{36}(what) \simeq 0.4$. In other words, the smaller network alone deals with the easier "where" task, while both networks effectively merge into one large network to deal with the harder "what task". In the full evolutionary simulations, with all three constituent networks, the results are variable, as we would expect from the flatness of the learning time landscape. Invariably, the two networks with hidden units combine themselves together to deal with the harder "what" task. Sometimes the "where" tasks is dealt with exclusively by the no hidden layer network, leaving a fully modular architecture. Sometimes, the 18 hidden unit network helps out with it. Sometimes, the no hidden layer network also helps out with the "what" task. The conclusion seems to be that the evolution of modularity due to learning rate advantage is not as inevitable as is often thought.

5.4 Temporal Cross-Talk

Jacobs et al.[6] suggested that modularity would minimise the negative effects of temporal cross-talk, i.e. the interference of learning and performing one task due to learning a different task at a different time rather than learning the two tasks at once. To investigate this, they used a different training regime.

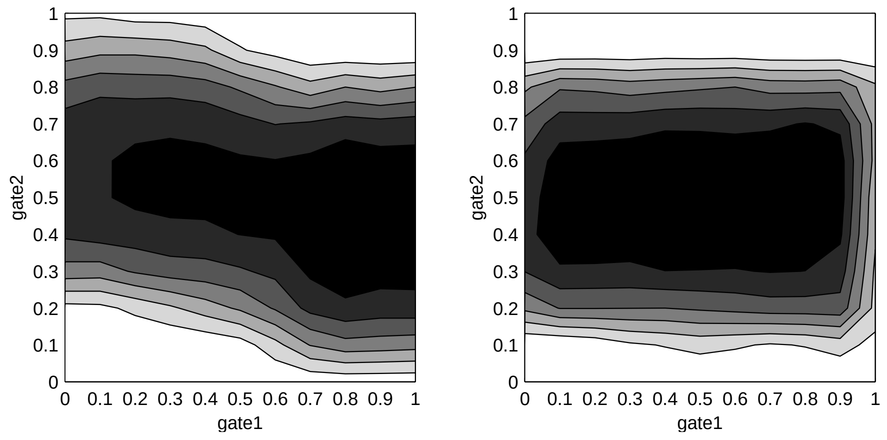

Figure 7. Learning Times in Temporal Cross-talk Simulations.

They compared the standard procedure (i.e. presenting all 162 training examples in random order) against presenting the examples in blocks with all 81 examples for one task together, and then all 81 examples for the other task together. Fully distributed networks learned significantly slower with the blocked training data, but the modular architecture of Figure 2 showed no difference.

Of all our gated networks, only the Pre-Sigmoidal Gated Networks with Cross Entropy cost function have evolved non-modular architectures, so it is only in that case we can expect using the blocked training data to make a difference. Figure 7a shows the learning times when we use the evolved learning parameters obtained with un-blocked training data. If we evolve the network to deal with blocked training data, the learning parameters adjust appropriately and the architecture dependence on the learning times end up as in Figure 7b. We actually end up with a less modular architecture in this case!

5.5 Spatial Cross-Talk

The final factor to consider is spatial cross-talk. If both tasks need to be performed simultaneously, then it is likely that cross-task between the tasks in a non-modular system will reduce the performance compared with the modular case. The gated model of Figure 2 can easily be modified to test this. All we need to do is remove the task inputs to the constituent networks, extend each output layer to 18 units, and have separate gates for the "What" and "Where" output units[6]. In this case, with Post-Sigmoidal gates, the gates will go binary as discussed in Section 5.3 and simple considerations of maximal use of hidden unit resources will inevitably result in a modular architecture. For Pre-Sigmoidal gates, we just end up with a re-parameterization of the Rueckl et al. type networks of Figure 1, and hence the same modularity dependence on learning algorithm.

6 Conclusions

We have investigated some of the main issues involved in simulating the evolution of gated networks and the emergence of modularity in them. In some cases it is easy to force the evolution of modularity simply by restricting the options open to the evolutionary simulations. In others, the evolution of modularity depends crucially on the choice between competing non-biologically plausible learning algorithms. Whilst it is easy to find reasons why modular neural systems should be superior to their non-modular counterparts, we have seen that simulating the evolution of such modularity is not as straightforward as it is often assumed.

References

1. Belew, R.K. & Mitchell, M. (Eds) (1996). *Adaptive Individuals in Evolving Populations.* Reading, MA: Addison-Wesley.
2. Bullinaria, J.A. (2001). Exploring the Baldwin Effect in Evolving Adaptable Control Systems. In: R.F. French & J.P. Sougne (Eds), *Connectionist Models of Learning, Development and Evolution*, 231-242. London: Springer.
3. Bullinaria, J.A. (2001). Simulating the Evolution of Modular Neural Systems. *Proceedings of the Annual Conference of the Cognitive Science Society*, 146-151. Mahwah, NJ: LEA.
4. Goodale, M.A. & Milner, A.D. (1992). Separate Visual Pathways for Perception and Action. *Trends in Neurosciences*, **15**, 20-25.
5. Hinton, G.E. (1989). Connectionist Learning Procedures. *Artificial Intelligence*, **40**, 185-234.
6. Jacobs, R.A., Jordan, M.I. & Barto, A.G. (1991). Task Decomposition Through Competition in Modular Connectionist Architecture: The What and Where Vision Tasks. *Cognitive Science*, **15**, 219-250.
7. Mishkin, M., Ungerleider, L.G. & Macko, K.A. (1983). Object Vision and Spatial Vision: Two Cortical Pathways. *Trends in Neurosciences*, **6**,

414-417.

8. Plaut, D.C. & Hinton, G.E. (1987). Learning Sets of Filters Using Back-Propagation. *Computer Speech and Language*, **2**, 35-61.
9. Rueckl, J.G., Cave, K.R. & Kosslyn, S.M. (1989). Why are “What” and “Where” Processed by Separate Cortical Visual Systems? A Computational Investigation. *Journal of Cognitive Neuroscience*, **1**, 171-186.
10. Shallice, T. (1988). *From Neuropsychology to Mental Structure.* Cambridge: Cambridge University Press.