UESMANN: a Feed-Forward Network Capable of Learning Multiple Functions

James C. Finnis and Mark Neal

Computer Science, Aberystwyth University, Aberystwyth UK (jcf1@aber.ac.uk, mjn@aber.ac.uk)

Abstract. A number of types of neural network have been shown to be useful for a wide range of tasks, and can be "trained" in a large number of ways. This paper considers how it might be possible to train and run neural networks to respond in different ways under different prevailing circumstances, achieving smooth transitions between multiple learned behaviours in a single network. This type of behaviour has been shown to be useful in a range of applications, such as maintenance of homeostasis. We introduce a novel technique for training multilayer perceptrons which improves on the transitional behaviour of many existing methods, and permits explicit training of multiple behaviours in a single network using gradient descent.

This work introduces UESMANN, a neural network which can smoothly switch between different behaviours using a very simple neuromodulatory paradigm. The network is currently trained using UESMANN-BP, a simple modification to the standard backpropagation algorithm[13, 9]. It is likely that heuristic search techniques will also prove successful, but these have yet to be explored. UESMANN's major advantages over naïve linear interpolation between outputs are a wider and more consistent "transition region" between behaviours, and the ability for intermediate points along the transition to also be trained.

We will not deal with applications in this paper, concentrating instead on simple functions to explore the system. However, smooth switching between learned functions is useful in situations where an embodied system's behaviour needs to be synchronised with changes in its environment, or where the behaviour needs to change over time. In such systems, a small change in the controlling parameter should result in a small change in the behaviour, along a continuum between the two trained behaviours: i.e. behaviour blending, not behaviour selection. One example is the maintenance of homeostasis, such as husbanding battery charge in a solar-powered robot. Another is striking a balance between exploration and exploitation of resources[10].

1 Background

Most artificial neural networks model electrical communications between neurons, but this is not the only communication neurons have. Neuronal behaviour

is also modified by chemicals which diffuse through the intercellular space: this is termed *neuromodulation*[5]. Our technique uses a simple model – perhaps the simplest possible – of neuromodulation in the weights of a multilayer perceptron.

Two other techniques which model neuromodulation are GasNets and Artificial Endocrine Systems (AES). In GasNets, the modulation is of the activation function in a recurrent network[4]. Topology, weights, modulator emission and sensitivity are trained using genetic algorithms. They are useful in solving problems with temporal elements[3, 11], and prove more evolvable than the CTRNN[1, 7], perhaps the most common recurrent network used in robotics. This may be due to the loosely-coupled co-evolution of two communications channels[12].

AESs are inspired by the endocrine system, modelling glands releasing hormones in response to environmental or internal changes. In an AES, the modelled substances modulate multilayer perceptron weights[8]. The network is typically trained with backpropagation to perform a task, and the modulation implemented "by hand", uniformly across the entire network[8] or on a single layer[10]. Engineering the hormone in this way, rather than allowing the hormone to be evolved (as in a GasNet), provides deterministic behaviour and allows the designer to focus the time-dependent aspect of the behaviour on a particular part of the problem. AESs have applications in stress response[8] and homeostasis[10].

Both GasNets and AESs provide a modulator release, saturation and decay model. UESMANN currently does not, concentrating only on the modulation itself. However, the AES hormone model could be used.

2 Motivation

The core of the GasNet and AES models is the modulation of a neural network by a global parameter decoupled from the network proper, allowing the network to respond to its environment. This response typically takes the form of a smooth transition between modes of operation. However, it is difficult to design systems which can move smoothly between qualitatively different learned behaviours. GasNets, NSGasNets and CTRNNs rely on evolutionary search to find an overarching behaviour which provides the required sub-behaviours (and which may contain undesired emergent behaviours). Current AES implementations simply generate less or more behaviour dependent on the hormone level: although the system is capable of considerably more, it is difficult to design.

For many applications it is useful to construct a system which is explicitly trained for two behaviours, with the modulator providing a smooth switch between them. It would also be useful if the network could be trained to behave a certain way at intermediate modulator levels – that is, if some control over the transitional behaviour were available.

The present work demonstrates a neuromodulatory technique which can be so trained (currently using a supervised learning technique), and which can sensibly interpolate between trained behaviours for intermediate values of the modulator.

The technique effectively creates a single feed-forward network whose weights are modulated. As such, each of the functions it performs cannot be temporal in nature, although the entire system may be if the modulator has a temporal element. This is essentially true of GasNets and AES: the networks themselves and their function at any given time are are atemporal, but the modulation adds the temporality.

3 The UESMANN-BP Algorithm

The UESMANN algorithm is based on the AES given in [10], but all weights have equal sensitivity to the modulator. This "simplest possible" form of neuromodulation consists of a network in which each node has the function

$$a_i^l = \sigma\left(b_i^l + \sum_j (h+1) w_{ij}^l a_j^{l-1}\right) \tag{1}$$

where a_i^l is the activation of node i in layer l, b_i^l is the bias of node i in layer l, and w_{ij}^l is the weight of the connection between node j in layer $l-1$ and node i in layer l. The activation function σ is a sigmoid (we use the logistic function).

Each weight is modulated by a "hormone" parameter h such that $h < 0$ inhibits the connection, and $h > 0$ excites it. In the current work $0 \leq h \leq 1$, so the hormone is always excitatory.

Given that we train the network for different functions $h = 0$ and $h = 1$, the initial $h = 0$ function will have the weights take their nominal values, while the $h = 1$ function will have the weights effectively doubled. Thus, our algorithm should find a set of weights and biases $(\mathbf{b}, \mathbf{w})$ which performs one function, while $(\mathbf{b}, 2\mathbf{w})$ performs the second. The training algorithm we shall use is dubbed UESMANN-BP, and is a supervised learning algorithm based on alternating backpropagation for each modulator level. We train the weights and biases $(\mathbf{b}, \mathbf{w})$ for the first function, and alternate this with training the weights and biases $(\mathbf{b}, 2\mathbf{w})$ for the second function. Thus, in each training iteration the weights will move towards a solution for the first function, then towards a solution for the second. This is shown in Algorithm 1.

Because backpropagation is done by calculating the cost gradient with respect to the weight, which is effectively $(h+1)w$, we must adjust the algorithm accordingly. This gives the following equations:

$$\frac{\partial C}{\partial w_{ij}^l} = (h+1) a_j^{l-1} \delta_i^l \qquad \text{error surface gradient wrt. weight} \tag{2}$$

$$\frac{\partial C}{\partial b_i^l} = \delta_i^l \qquad \text{error surface gradient wrt. bias} \tag{3}$$

$$\delta_j^L = (a_j^L - y_j) \cdot a_j^L \cdot (1 - a_j^L) \qquad \text{error in output layer} \tag{4}$$

$$\delta_j^l = a_j^l (1 - a_j^l) \sum_i (h+1) w_{ij}^{l+1} \delta_i^{l+1} \,. \qquad \text{error in hidden layer} \tag{5}$$

where C is the quadratic cost function of the output layer and y_j is the required value of output j. These are the standard backpropagation functions modified so that the cost gradients are now with respect to $w(h+1)$. As in standard backpropagation, we repeatedly run the network forwards, calculate the error, and add (given an learning rate η) $\eta \frac{\partial C}{\partial w_{ij}^l}$ to each weight and $\eta \frac{\partial C}{\partial b_i^l}$ to each bias.

4 Methodology

What UESMANN attempts to do is novel: generate a multilayer perceptron with two (or more) functions, such that the functions are smoothly switched between as the modulator varies. Therefore it does not directly compare with any of the existing neuromodulatory systems, or with CTRNNs. However, we can compare it with other ways of "morphing" between two feed-forward multilayer perceptrons. The two other methods we shall evaluate are linear interpolation between the outputs of two networks (which we term "output blending"), and linear interpolation of the weights and biases of two networks ("network blending"). These are selected because they are the most obvious and straightforward techniques for generating the required behaviour.

We will not be concerned with how the modulator is released, nor how it decays, simply how it affects the network. We wish to find out how well our system behaves in transition: how wide the transition region is, how useful it is, and how consistent the behaviour is across multiple training runs on the same data. Therefore we shall run the three techniques multiple times, and qualitatively examine the results. We have chosen classification problems as examples of typical problems for which perceptrons are used, in which a "transition region" might seem meaningless. However, the assigned classifications should shift gradually between the two behaviours over a large hormone range. In all cases, parameters such as learning rate and initial weights were determined after informal experimentation.

Our first experiments show the behaviour of UESMANN-BP attempting to learn pairings of the logical boolean connectives, chosen as the simplest possible binary functions. We shall then investigate the transition region widths

Algorithm 1 Training UESMANN for two functions using backpropagation. Each example is (in, out_1, out_2) where in is the input vector, out_1 and out_2 are the output vectors for the two functions, and a^L is the network output.

repeat
  for all examples (in, out_1, out_2) **do**
    present input in and run the network
    update $(\mathbf{b}, \mathbf{w})$ using UESMANN-BP with $a^L = out_1, h = 0$
    present input in and run the network
    update $(\mathbf{b}, \mathbf{w})$ using UESMANN-BP with $a^L = out_2, h = 1$
  end for
until converged for both functions or training limit reached

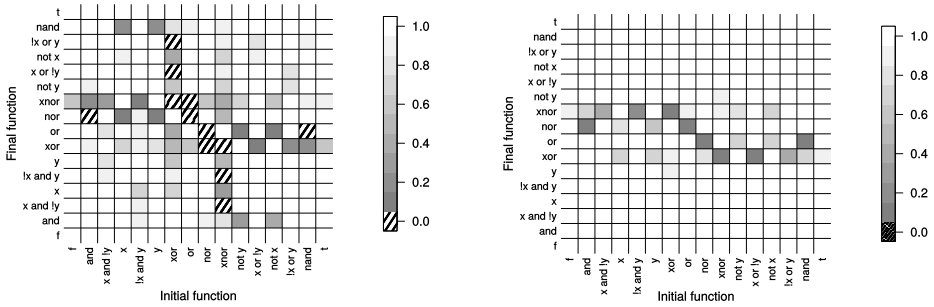
for a particular pairing (XOR and AND). We will then attempt to transfer UESMANN-BP to a real problem: smooth switching between two classifications of handwritten digits. The transition widths will be compared against output and network blending for different hidden node counts.

Finally, an attempt will be made to train a single network at three levels of modulator. Two different intermediate functions (for modulator 0.5) will be compared to see how the choice of intermediate function affects the convergence behaviour. For this test, a new problem will be used: detecting horizontal or vertical lines which may or may not appear in a noisy image. This is chosen both to demonstrate a new domain, and because it is easier to evaluate and classify intermediate functions.

5 Experimental Results

5.1 Logical Connectives

Our first experiment is designed to show whether UESMANN-BP is able to learn pairings of a wide range of functions. 400 runs of the algorithm were performed in networks with 2 and 3 hidden nodes. Weights and biases were initialised to uniform random numbers in the range [-0.5,0.5]. The system was given up to 100000 iterations to converge to an output layer error of < 0.05 for both functions, with a learning rate $\eta = 1$. The results, showing how many of the runs converged for both functions, are shown in Fig. 1.



(a) 2 hidden nodes (b) 3 hidden nodes

Fig. 1. Proportion of 400 runs which converged to a network able to perform both functions in all possible pairings of logical connectives. Two different hidden node counts were used. Cross-hatched squares mark pairings where no solution was found.

The results for the simplest 2-2-1 network are good, but certain pairs of functions will not converge to a network which produces both functions. Intuitively, these functions seem to be those for which it is hard to conceive of a useful intermediate stage (a large Hamming distance between the truth tables is a factor),

although the exact relationship needs to be determined. It is interesting that a network with 2 hidden nodes – the minimum number for linearly inseparable functions such as XOR – is able to learn two functions in a single network.

These difficulties remain but are surmountable when the hidden layer is increased to 3 nodes, although for some pairings the number of convergent runs is low. No analysis of the transition regions – either their size or functions – was performed.

Using the same η and initialisation range parameters, the transition behaviour of 2-2-1 UESMANN-BP networks was compared with output blended and network blended (see Sec. 4) networks for the pairing XOR→AND (i.e. with a zero modulator perform XOR, then transition to AND when the modulator is 1). 12 randomly initialised converging runs were performed of each, and the function performed by the network when the output is thresholded at 0.5 was plotted at each modulation value. The results are shown in Fig. 2.

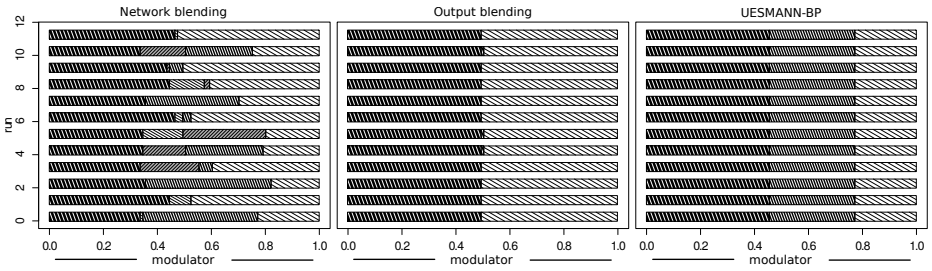


Fig. 2. Examples of different transition behaviours in output blended, network blended, and UESMANN-BP networks. The shading represents the actual function being performed by the generated network at the given modulator value, when the output is thresholded at 0.5.

Output blending produces almost no transition, with the network behaviour switching at 0.5. Network blending produces wider transitions, which appear random. This is probably because two unrelated networks are being blended, leading to a "competing conventions" problem. UESMANN-BP produces a wide, predictable transition consisting of the false function: a compromise between XOR and AND.

6 The MNIST Database

Having established that UESMANN-BP can generate networks which perform two different functions with a wide and predictable transition in simple cases, we now move on to a more difficult problem: the MNIST handwriting recognition database. This database has become a *de-facto* test problem for image classifiers, and consists of 70000 handwritten digits, divided into a training set of 60000 and a test set of 10000 examples. Each example consists of a 28×28 8-bit

monochrome image of a digit, size normalised and centered, with its associated label (the numerical value of the digit) [6].

Our experiment aims to learn two different labellings of MNIST examples at different network node counts. We wish to establish firstly that UESMANN-BP can learn such a complex pair of functions, secondly examine the transition behaviour of such a network, and thirdly explore the effect of node count in comparison with a control (i.e. a plain backpropagation network learning only one labelling).

In our networks there are 784 inputs, one for each pixel normalised to the range [0,1], and 10 outputs. In each example, the output corresponding to the correct label is set to 1, while the others are set to 0. During testing, the index of the highest output is considered to be the result.

The basic training algorithm is the same as for the previous experiments: each example is presented first for the function with $h = 0$, then for the function with $h = 1$, with different outputs for each function being learned. The examples are iterated over in the order they appear in the training set. In these tests, the initial weights and biases were set by Bishop's rule of thumb[2]: each value is in the range $\left(\frac{-1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right)$, where d is the number of inputs to the node. It is likely that more tests should be done to determine an optimal value.

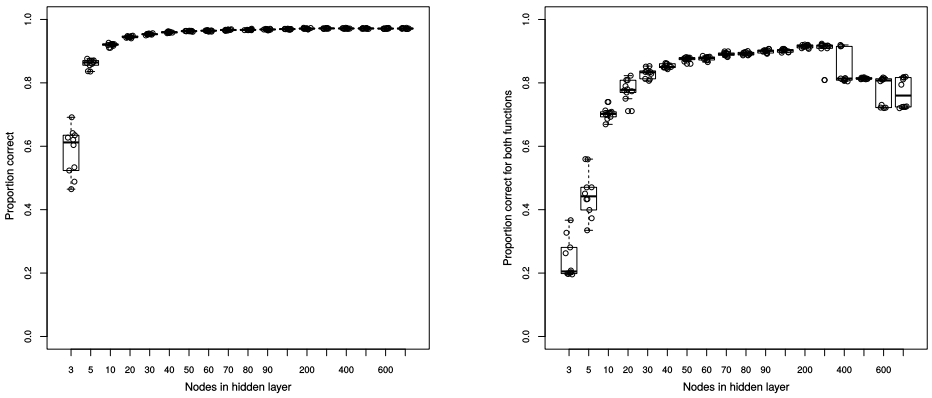
Convergence was measured by holding a validation set of 10000 examples out from the training set, and using a small slice from that set every 300 iterations. The resulting network, after 300000 iterations, is the one which performed best in all validations throughout training. Once trained, the test set is used to evaluate this best network.

The two mappings to be learned are the nominal mapping, where each image is assigned the label it has in the database; and a mapping in which adjacent labels are swapped – i.e. a image showing "0" will set output 1 high, while a "1" will set output 0 high and so on. The Hamming distance between the functions is maximal: for no input do the two functions give the same result.

The results are shown in Fig. 3. In the control, peak performance is achieved at 400 nodes with a 2.6% error rate, but all node counts over 30 give at worst a 5% error rate. This roughly agrees with the performance reported by LeCun et al. in [6].

For UESMANN-BP, which learns two functions in one network, the best performance is achieved at 300 nodes, with a 7.6% error rate — i.e. the best network for the best run at 300 nodes correctly performed both functions for all but 7.6% of the examples. After this, the ability of the network to find a solution rapidly falls, suggesting poor local minima.

The transition behaviour of the best network was reconstructed in Table 1. We can see that the transition starts at around 0.25 and ends at 0.5, with values generally changing separately, giving a smooth transition. To compare the transition regions, 10 training runs were performed with all three methods. These were then run at 100 different modulator values on the test set. The sizes of the transition regions obtained are shown in Fig. 4.



(a) Control performance for single function (backprop)

(b) Performance for learning two functions in one network (UESMANN-BP)

Fig. 3. The means of the success rate at both functions by validation for all runs at each node count for both control and UESMANN-BP.

Table 1. Most common output given the first 1000 MNIST examples passed to the best 300-node network generated using UESMANN-BP, at each value of h. The transition region is marked with vertical lines. For clarity, repeated values are shown with dots.

h	**0.00**	**.05**	**.10**	**.15**	**.20**	**.25**	**.30**	**.35**	**.40**	**.45**	**.50**	**.55**	**.60**	**.65**	**.70**	**.75**	**.80**	**.85**	**.90**	**.95**	**1.00**
Label 0	**0**	·	·	·	·	·	·	·	**0**	1	·	·	·	·	·	·	·	·	·	·	1
Label 1	**1**	·	·	·	·	·	·	·	·	**1**	0	·	·	·	·	·	·	·	·	·	0
Label 2	**2**	·	·	·	·	·	·	·	·	·	**2**	3	·	·	·	·	·	·	·	·	3
Label 3	**3**	·	·	·	·	·	·	**3**	2	·	·	·	·	·	·	·	·	·	·	·	2
Label 4	**4**	·	·	·	·	·	·	**4**	5	·	·	·	·	·	·	·	·	·	·	·	5
Label 5	**5**	·	·	·	·	·	**5**	4	·	·	·	·	·	·	·	·	·	·	·	·	4
Label 6	**6**	·	·	·	·	·	·	·	**6**	7	·	·	·	·	·	·	·	·	·	·	7
Label 7	**7**	·	·	·	·	·	**7**	6	·	·	·	·	·	·	·	·	·	·	·	·	6
Label 8	**8**	·	·	·	·	·	·	**8**	9	·	·	·	·	·	·	·	·	·	·	·	9
Label 9	**9**	·	·	·	·	**9**	8	·	·	·	·	·	·	·	·	·	·	·	·	·	8
Behaviour	Function 1						Transition					Function 2									

Output blending clearly produces an abrupt transition, while network blending produces transitions whose width depends on the hidden node count. However, as has already been discussed, the transitional networks produced in the latter are likely to be of little use, because of competing conventions. For UESMANN-BP runs where both functions converge, the transition width is consistent, at about a third of the modulator range.

Thus, UESMANN-BP is capable of producing a consistent and potentially useful smooth switching behaviour between two classifications in a real-world problem. Although output blending will converge to better solutions, it produces much more abrupt transition regions and cannot be trained at intermediate values (which is possible with UESMANN, as we shall see).

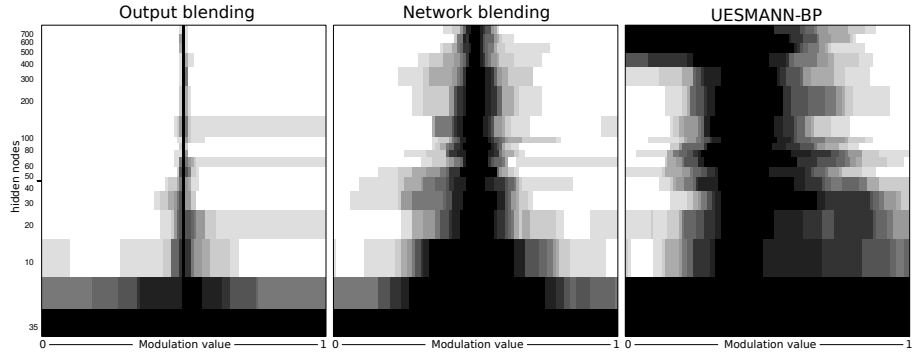


Fig. 4. The transition zones for networks learning two different MNIST mappings using output blending, network blending and UESMANN-BP. 100 examples were run for each network at 100 different morph points, and the networks were generated by 10 different attempts at learning at each node count. Brightness shows how many of the attempts are performing one of the two learned functions, rather than some intermediate.

7 Training at Intermediate Modulation Values

While the behaviour demonstrated thus far is useful, it gives no control over the transitional region. It may be possible to train a different function for a third, intermediate modulator value to provide this control. It seems likely that it will be considerably more difficult to find a solution when the intermediate function is not a "natural" intermediate between the two end functions.

The MNIST handwriting recognition set is not ideally suited for experiments here, because it is hard to identify a clear intermediate between two functions at all 10 outputs. A simpler case is needed, with a single output. Therefore a new experiment was devised: recognising horizontal or vertical lines in an image.

This experiment uses data in the same format as the MNIST data. Each image contains either a horizontal line, a vertical line or no line, labelled 0, 1 and 2 respectively. Each image was overlaid with Gaussian noise ($\mu = 0.4, \sigma = 0.15$) and blurred with a 3×3 Gaussian kernel. The data set sizes were the same as for the MNIST database, with equal proportions of each image type provided. The networks trained had a single output, indicating whether a line of the appropriate orientation was detected or not. A representative sample of the images is shown in Fig. 5. In all experiments, the initialisation values

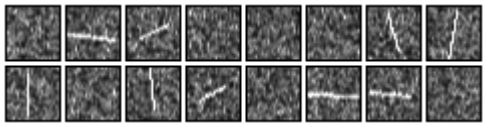


Fig. 5. Examples of images used for line detection experiments.

were determined using Bishop's rule, and $\eta = 0.05$: a lower η was required for convergence when intermediate values were used.

Three experiments were performed: in the first, no intermediate training point was given. In the others, two different intermediate functions were trained at modulator $h = 0.5$: the first detects any line (i.e. output is 1 when either a vertical or horizontal line is present), the second detects no lines (i.e. output is 1 for blank images). 5 runs were performed for a limited subset of the hidden node counts used in previous experiments, and each network analysed against the test set. The results are in Fig. 6.

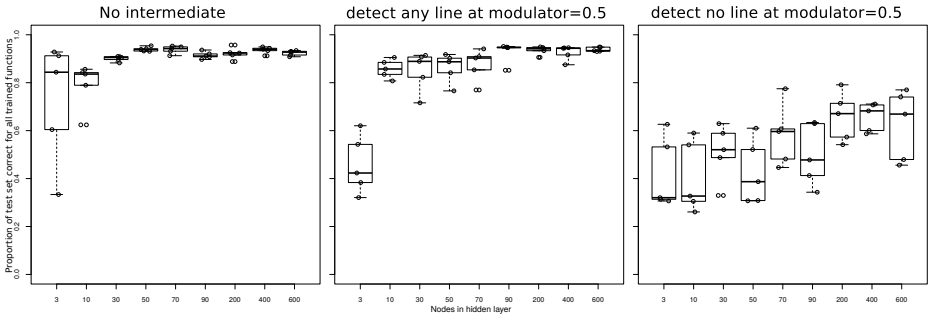


Fig. 6. Performance of line detection, with $h = 0$ detecting horizontal and $h = 1$ detecting vertical lines. Experiments show no intermediate function being trained, $h = 0.5$ detecting any line and $h = 0.5$ detecting no line.

Without an intermediate function peak performance is at 200 nodes with a 4.28% error, but even modest node counts can perform well: one 50 node network achieves 4.6% error, and even a 3 node network achieves 7.18% error. It is possible that more runs would produce significantly better results.

For additionally detecting either line when $h = 0.5$, best performance is 4.89% error at 90 nodes – i.e. for all but 4.89% of the examples, the network gives the correct output for all three learned functions. This clearly shows that it is possible to train at least this intermediate function to a high level of performance.

For detecting no line as the intermediate function, best performance is much worse at 20.88% error at 200 nodes, and convergence is slower by about an order of magnitude. This is to be expected, because we are moving from three functions which detect lines to two functions which detect lines and a third which detects no lines — there is now an implicit negation in the intermediate function. It seems harder to learn intermediate functions which are not some "natural" intermediate of the two end points.

To determine the "default" intermediate behaviour, the best 200-node network from the first experiment was run with each example from the test set, at three different modulator values (0, 0.5 ,1). The number of images detected with each of the three labels is shown in Table 2. Without a trained intermediate function, the network appeared to detect some (about a two thirds) of either

vertical and horizontal lines when $h = 0.5$. Looking at the images, it is hard to establish why certain lines are detected while others are not. However, the fact that the default intermediate behaviour is detection of both kinds of lines shows that this is a natural intermediate.

Table 2. Number of lines of each type detected by a network set to detect horizontal lines when $h = 0$, vertical lines with $h = 1$ with no intermediate function trained, at different modulator values.

Modulator	H-lines detected	V-lines detected	Blanks detected
0	3258	83	67
0.5	2122	2283	4
1	29	3221	77

8 Conclusions and Further Work

The UESMANN network is a simple modulated network which achieves smooth switching between qualitatively different functions in a consistent way, with a wide transition region. When compared with naïve linear interpolation between outputs of different networks, the transition region is much wider. Wide transition regions are desirable because they provide a smooth transition between the different functions, permitting adaptive behaviour where the modulator value (often from the environment) is between the two extremes.

Naïve blending of network weights and biases produces wider regions, but they are unpredictable and likely to contain "nonsensical" functions because of the competing conventions of the two parent networks. UESMANN transition behaviour tends to have functional "compromises" between the end-point functions, although this behaviour needs to be explored and analysed further. Providing the modulator as an additional input, rather than a global modulator, should also be tested: we predict that while this may converge well, it will result in narrower transition regions than UESMANN due to the shape of the sigmoid activation function.

A UESMANN network can be successfully trained using backpropagation as a supervised learning technique if suitable examples of both functions are available. The backpropagation parameters need to be investigated, such as initialisation range: preliminary work suggests that larger initial values often have more success. To provide consistent convergence it is likely that more hidden nodes should be used than are required for the "parent" functions. However, some runs do succeed without significantly more nodes.

While a typical UESMANN application may feature only two functions, the network can be trained to perform other functions at intermediate values of the modulator parameter. How successful this training is depends on the nature of these intermediate functions compared with the two end-point functions.

UESMANN and UESMANN-BP may prove useful in any application which requires smooth switching between qualitatively different learned behaviours. These include the maintenance of homeostasis in autonomous systems, power management and so on.

As well as deeper analysis of the parameters of the algorithm itself, further work includes building a UESMANN network into a robot for field tests of homeostatic behaviour, and using genetic algorithms for reinforcement learning of UESMANN networks (UESMANN-GA), with a suitable modulator release/saturation/decay scheme. These experiments will use continuous inputs and outputs, which should show the benefit of smooth, wide transition regions by avoiding the oscillation around the switching transition often exhibited by such systems.

References

1. Beer, R.D., Gallagher, J.C.: Evolving dynamical neural networks for adaptive behavior. Adaptive behavior 1(1), 91–122 (1992)
2. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press (1995)
3. Husbands, P., McHale, G.: Quadrupedal locomotion: GasNets, CTRNNs and Hybrid CTRNN/PNNs compared. In: Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems (ALIFE IX). pp. 106–112 (2004), http://sro.sussex.ac.uk/16037/
4. Husbands, P., Smith, T., Shea, M.O., Jakobi, N., Anderson, J., Philippides, A.: Brains, Gases and Robots. ICANN'98:Proceedings of the 8th International Conference on Artificial Neural Networks, Skövde, Sweden, 2-4 September 1998 (Perspectives in Neural Computing) pp. 51–63 (1998)
5. Kaczmarek, L.K., Levitan, I.B.: Neuromodulation: the biochemical control of neuronal excitability. Oxford University Press, New York (1987)
6. Lecun, Y., Cortes, C.: The MNIST database of handwritten digits http://yann.lecun.com/exdb/mnist/
7. Magg, S., Philippides, A.: Gasnets and ctrnns–a comparison in terms of evolvability. In: From Animals to Animats 9, pp. 461–472. Springer (2006)
8. Neal, M., Timmis, J.: Timidity: A Useful Emotional Mechanism for Robot Control? Informatica (Slovenia) 27(2), 197–204 (2003)
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
10. Sauze, C., Neal, M.: Artificial Endocrine Controller for Power Management in Robotic Systems. Neural Networks and Learning Systems, IEEE Transactions on 24(12), 1973–1985 (2013), http://dx.doi.org/10.1109/TNNLS.2013.2271094
11. Smith, T., Husbands, P., Philippides, A., O'Shea, M.: Neuronal plasticity and temporal adaptivity: GasNet robot control networks. Adaptive Behavior 10(3-4), 161–183 (2002)
12. Vargas, P.A., Di Paolo, E.A., Husbands, P.: A study of GasNet spatial embedding in a delayed-response task. In: Artificial Life XI, Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems. pp. 640–647 (2008)
13. Werbos, P.: Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard (1974)