Connectionist Simulation of Quantification Skills

Khurshid Ahmad, Matthew Casey and Tracey Bale

Centre for Knowledge Management, Department of Computing, School of Electronics and Physical Sciences, University of Surrey, Guildford, Surrey, GU2 7XH, UK.

Correspondence and offprint requests to: Professor K. Ahmad, Centre for Knowledge Management, Department of Computing, School of Electronics and Physical Sciences, University of Surrey, Guildford, Surrey, GU2 7XH, UK. Tel: 01483 689322. Fax: 01483 686051. E-mail: k.ahmad@surrey.ac.uk.

Running head: Multi-net Simulation of Quantification

Keywords: Multi-net, modularity, mixture-of-experts, gating neural networks, recurrence, quantification, numerosity discrimination, subitizing, counting.

Contents

Abstract

The study of numerical abilities, and how they are acquired, is being used to explore the continuity between ontogenesis and environmental learning. One technique that proves useful in this exploration is the artificial simulation of numerical abilities with neural networks, using different learning paradigms to explore development. A neural network simulation of subitization, sometimes referred to as visual enumeration, and of counting, a recurrent operation, has been developed using the so-called multi-net architecture. Our numerical ability simulations use two or more neural networks combining supervised and unsupervised learning techniques to model subitization and counting. Subitization has been simulated using networks employing unsupervised self-organising learning, the results of which agree with infant subitization experiments and are comparable with supervised neural network simulations of subitization reported in the literature. Counting has been simulated using a multi-net system of supervised static and recurrent backpropagation networks that learn their individual tasks within an unsupervised, competitive framework. The developmental profile of the counting simulation shows similarities to that of children learning to count and demonstrates how neural networks can learn how to be combined together in a process modelling development.

1 Introduction

There is considerable discussion in the literature about how human knowledge of numbers and number systems develops and to what extent such knowledge is, in some ways, innate. Brannon (2002), in her review of the development of "numerical knowledge", argues that such development shows evolutionary and ontogenetic continuity; the latter being the more controversial proposal. Evolutionary continuity relates to the literature on how number systems are represented by adult humans and animals whereas ontogenetic continuity deals with the numerical abilities of infants and how these abilities grow into a numerate adult (2002:223-224).

Brannon concentrates upon ordinal numerical knowledge, including ordinal relations between numerical values. Here, the distinction is drawn between ordinal relations which allow an understanding that, for example, 16 is numerically more than 8, but that ordinal numerical knowledge is required to understand the ordinal direction of a sequence of ordered numbers. Her findings suggest that infants as young as 11 months of age are sensitive to ordinal relations but that this does not necessitate ordinal numerical knowledge. This is in contrast to Dehaene and Changeux's (1993) prediction, based on an artificial neural network (ANN), which suggests that such knowledge develops between 9 and 11 months of age.

The use of ANNs to model numerical abilities allows psychologists to explore models of numerical processing systems in comparison with observational data, especially from developmental and disability data related to numerical evolution and ontogenesis (see Nye et al 1995, Butterworth 1999 and Dehaene 2000). Mareschal and Johnson cite the use of such computational models to 'provide rigorous and tangible accounts of development' (2002:154). Here 'the modeler is forced to make explicit what is meant by 'representation', 'acquired knowledge' [and] 'innate knowledge'', to provide 'a set of possible solutions' (2002:154-155). These possible solutions are also themselves the cause of debate and can sometimes cloud the core issues, as noted by

Cohen and Chaput who ‘admit that [their] own attempts [to model cognitive development] influence [the] view of other models’ (2002:173).

Despite the apparent problems, modelling numerical abilities is important. For instance, ANNs have been used extensively to model quantification skills, including subitization, the so-called phenomenon of the discrimination of visual number, and counting, the ability to impose order on a set of objects by virtue of an abstract number system. However, whilst these models have generated a number of different avenues of research (see, for example, Dehaene 2000), they have not tended to directly address an understanding of the continuity between ontogenesis and environmental learning. Instead, the majority have either modelled innateness by ignoring learning (Dehaene and Changeux, 1993) or have allowed systems to develop through a process of external tutoring, a supervised learning technique that can be likened to environmental learning (Mareschal and Shultz 1999, Rodriguez et al 1999, Peterson and Simon 2000). Earlier on, Dehaene suggested that experiments on children have shown that ‘the truth […] seems to stand somewhere between the “all innate” and “all acquired” extremes’ (1997:119), but there has been little done to explore this with ANNs. These relationships can be explored through ANN techniques such as unsupervised learning, where a system can undergo a process of learning without a teacher. Furthermore, a combination of both supervised and unsupervised techniques can be achieved through multi-net systems, allowing further exploration of the continuity between ontogenesis and environmental learning.

In this paper, we review different models of subitization and counting, from earlier experiments that explored numerically related abilities through to models that deal directly with specific numerical processing. The majority of these models either deal with development through hard-wired connections or through learning with a teacher. We conclude that techniques using unsupervised learning warrant investigation, and go on to discuss two models that we have developed as a consequence of this: one simulates aspects of subitization and one counting. Both use unsupervised learning within multi-net systems to demonstrate how different types of learning can be used to explore the relationship between ontogenesis and environmental learning. Our systems perform as well as, if not marginally better than, other reported simulations, such as Peterson and Simon (2000), Amit (1988) and Dehaene and Changeux’s (1993), with our results comparing favourably with that observed in children.

1.1 Subitizing: Numerical Discrimination and Visual Enumeration

Two early studies throw some light on the phenomenon of subitization: Jevons suggested that subitization is related to ‘the power of numerical discrimination’ (1871); Kaufman et al talked about discrimination of visual number (1949). More recently, Peterson and Simon define subitizing as ‘the process associated with enumeration of small collections’ (2000:94). The results of systematic observations on humans suggest that whilst the time it takes to visually enumerate objects lengthens as the number of objects in a typical visual scene increases, there is a discontinuity in the rate of change of reaction time with the number of objects – up to 50 msec/object for scenes containing up to three objects rising to 250-300 msec/object for scenes containing four or more objects (see, for example, Trick and Pylyshyn 1994). Furthermore, the accuracy of visual enumeration or subitization decreases as the number of objects increases, and again the decrease is accelerated for higher numerosities. This suggests that the subitization task is subject to a ‘subitizing limit,’ that is visual enumeration is limited to smaller numerosities of up to three or four, and that 'subitizing slope' shows a marked change for higher numerosities.

There are two major strands of thought in psychology about the nature and function of subitization. The first strand can be dated to the publication of a paper by Kaufman et al (1949) who suggested that subitizing is the apprehension of numerosity immediately upon presentation, without the need to resort to any form of counting. This view has been criticised by Gelman and Gallistel (1978) who argued that subitization is a form of preverbal counting. The second strand of thought has its origins in the work of Mandler and Shebo (1982). They have argued that adults *learn* to subitize through the recognition of canonical patterns. There is substantial opposition to this view. For example, Wynn (1995) has argued that the ability to subitize is inborn and hence independent of any number system (see also, Strauss and Curtis, 1984; Sophian and Adams, 1987; Fuson, 1988; Wynn 1992a, b; Kirby, 1992 and Spelke, 1994).

1.2 Counting: Recurrence and the Imposition of Order

Counting requires an analysis of the visual scene to ascertain the presence or absence of objects irrespective of their size. Once a size invariance has been established, counting requires some other capabilities of a numerosity system in contrast to subitization. This difference can be elaborated by the earlier observations of Donald Hebb, and their more recent refinement by Daniel Amit (1989), on the topic of counting identical chimes to tell the hour; identical chimes are the acoustic equivalent of size invariant objects abstracted from a visual scene. According to Amit such counting has 'several dimensions' (Amit 1989:241-245) which will form the basis of a 'connectionist' counting system: (i) the system should be able to *recognise* the arrival of several identical stimuli through 'some short term memory mechanism that does not transform into long term memory'; (ii) the system should be able to *identify* 'a generic chime in order to provoke counting'; and, (iii) the system should be able to *discriminate* 'between different temporal sequences of chimes, according [to] the abstract property which is their cardinal number' (Amit 1989:241). There is a suggestion that in order to recognise the 'several identical stimuli' one needs to employ recurrent networks. The ability to identify and discriminate are learnt with the help of a tutor who can provide training in how to start and to continue counting and how to assign the cardinality to the visual scene (or acoustic input) as a whole.

For some authors, counting is a learnt process by which an accurate value for the numerosity of a set of items can be determined through the use of a serial set of rules and short-term memory, allowing a pairing of objects with numeric labels from a number system. Gelman and Gallistel (1978) proposed the five 'how-to-count' principles by which counting can be defined: one-to-one correspondence, stable order, cardinality, abstraction and order irrelevance. Gelman and Meck (1983) identified the first three of these principles as the counting procedure, the fourth as to which types of set counting can be applied, and the fifth distinguishes counting from labelling. Fuson et al (1982) and Fuson (1988) investigated the types of errors that are made by children when counting. *Word errors* produced during the counting sequence are characterised within three sections: stable and conventional, stable and non-conventional, and unstable and non-conventional. Additional word errors concern the reduced frequency of production of irregular number words, and the strong association between a number word and the two previous words. Fuson also identified the two main *pointing errors* made by children as 'object skipped', where a child will miss pointing at an object, and ‘multiple count’, where a child will point at an object more than once.

1.3 Multi-net Architectures

The simulation of the development of numerosity in humans, from subitizing in infancy, through to counting in childhood, and the retention of both abilities in later life, appears an important challenge for modular artificial neural network systems. The challenge lies in the selection, training and the subsequent testing of the different methods of combining artificial neural network modules; these *multi-net* methods include the four proposed by Sharkey (1999): *co-operative*, *competitive*, *sequential* and *supervisory*.

Co-operative systems allow a group of networks to co-operate to produce an output, for example through the averaging of all the outputs. Competitive systems allow groups of networks to compete for the right to process a particular input signal, and can subsequently be trained to better respond to that signal in a *winner-take-all* fashion. Sequential systems allow the output of one network to become the input to another, allowing a chain of processing to be performed. Supervisory systems look at how learning can be improved through the use of a network that can be taught optimum learning parameters, which can then be applied to improve learning in different networks.

We propose that the constituent networks of a modular ANN may have to be trained using algorithms for unsupervised learning, particularly for simulating subitizing, and using algorithms for supervised or reinforcement learning, especially for simulating counting. Supervised learning can be compared with the effects of the environment over the organism, where environmental input supervises learning. Unsupervised learning can be compared with an organism’s built-in self-motivation. Within the multi-net methods described by Sharkey, networks employing both unsupervised and supervised learning in a modular system can be used in competitive and sequential configurations. Relating this to the simulations presented in this paper, we can see that counting involves learning a number system and associated number words, with some sense of addition and subtraction. The number system is not acquired spontaneously but appears to be taught. Conversely, it is not intuitively possible to think that subitization could be taught. These contrasting approaches helped us to define a simulation framework in which both supervised and unsupervised techniques are important.

We start this paper with a review of the different simulations of numerical abilities that have been performed, both using single network systems and multi-nets (section 2). Next we describe how modular multi-net neural systems can be trained to simulate subitization and to simulate counting. The subitizing system (section 3) can apprehend numerosity of up to 5 objects fairly accurately; this system uses a sequential multi-net system including two Kohonen (1997) Self Organising Maps (SOMs), one for representing magnitude of numbers and the second to articulate numerosity – the two SOMs are linked to each other through a Hebbian network. Our system performs as well as, if not marginally better than, other reported simulations of subitization by Peterson and Simon (2000) who used a backpropagation network. Our counting system (section 4) is a competitive gated multi-net system that uses a recurrent backpropagation network for learning, using a short-term memory to keep track of counting without transforming this information into a long-term memory.

The counting system uses two gates: one feedforward network to point to the next object and the other to articulate the count. The system performs as well as Amit's neural attractor system (1988) for counting chimes and as well as Dehaene and Changeux's (1993) for counting objects in a visual scene. Section 5 presents the conclusions. First, the our simulations show that if there are dedicated 'cortical territories' in the brain for apprehending numerosity, they might comprise a distributed system whose components deploy a range of learning algorithms. Second, we show that one artefact of artificial neural simulation of subitization is the so-called edge effect: the lowest and highest numerals in a sequence of displays with different numerosities are learnt more accurately than all others. Third, we show that better methods are required to deal with the visual scene particularly the way in which the scene is abstracted into a collection of objects independent of the size and location of the object.

2 'Neuronal' Simulations of Subitization and Counting

Studies on both animals and humans have attempted to demonstrate that basic numerical abilities are biologically based and consequent of evolutionary processes (see, for example, Thompson et al 1970, Meck and Church 1983, Wynn 1995, Brannon and Terrace 1998). Dehaene's experiments (2000, 1997) highlight that when comparing the numerosity of sets of objects, both humans and animals encounter *distance* and *magnitude* effect phenomena. The fact that the greater occurrence of errors found when comparing numbers that are close together in magnitude as opposed to further apart is known as the distance effect, and the magnitude effect is the drop in performance observed when comparing numbers that are equal in distance, but have larger magnitudes. In addition, Fechner's law states that the perceived intensity of a number stimulus is proportional to the logarithm of the actual intensity, hence the internal representation of number is compressed at higher magnitudes.

Numerical processing has also been studied by comparing the behaviour of randomly selected persons with those of brain-damaged patients (see Dehaene 2000; Butterworth 1999 and references therein for details). These neuro-cognitive and neuro-biological studies have lead to the formulation of several models of processing basic numerical ability. Such abilities include quantification, transcoding and calculation, and require different representations of number for different tasks, including abstract, written and auditory. From these contrastive studies one is led to conclude that numerical processing takes place in specific areas in the *parietal* and *temporal lobes* of the brain. In the parietal lobe there is a cluster of neurons that is used to represent the magnitude of quantities observed. The magnitude representation cluster or network interacts with another distinct cluster that is used to store (and retrieve) the visual number form; this network is located in the temporal lobe. The *arithmetic facts* are articulated through the use of yet another cluster in the left hemisphere – the so-called verbal system. This tripartite model, also referred to as the triple code model, comprising networks for representing magnitude, encoding visual number form and for articulating arithmetic facts, is originally due to Dehaene (1997).

A number of other models have been proposed for understanding numerical processing in the human brain. Different areas have been suggested where the written, auditory and abstract forms of cardinal numbers are represented (see, for instance, McCloskey et al, 1985; and Cipolotti and Butterworth, 1995). Whilst it is important to investigate further these claims about this specific brain-areas hypothesis, such an assumption is of significant import for simulations of numerical processing.

These models propose how the quantification tasks of subitizing and counting might be split into simpler subtasks. For example, Dehaene (1992) encapsulates subitizing within the abstract number module, and counting within the verbal module.

There have been a number of neural network based simulations of arithmetic fact learning reported in the early 1990s. The factor common to a significant number of studies was that numbers were shown using an analogue number representation; there were reports of systems that used a hybrid system, including analogue and symbolic forms. Table 1 briefly describes the architecture of these systems together with the learning algorithm that was used. McCloskey and Lindemann (1992) and Dallaway (1994) used an analogue representation of number, whereas Anderson et al (1994) used a hybrid approach with both an analogue and symbolic representation together. Anderson et al argue that this permits a simple shift between the symbolic patterns and their corresponding magnitudes, since the two become associated with each other. The use of a recurrent network for a system that learns to count appears intuitively suited to the task: counting involves the use of local memory and recurrent systems include such concepts.

Table 1: Models of arithmetic fact learning and performance showing the learning systems and number representations they employed.

System	Task	Learning System	Number Representation	Reference
		Supervised Systems		
MATHNET	Learning	Backpropagation	Analogue	McCloskey and Lindemann (1992)
Dallaway	Learning and Performance	Backpropagation (with output activation equation)	Analogue	Dallaway (1994)
		Unsupervised Systems		
Hybrid	Learning	Brain-State-in-a-Box (recurrence)	Analogue and symbolic	Anderson, Spoehr and Bennett (1994)

In the following sections we describe neural network based simulations of subitization (section 2.1) and of counting (section 2.2) to examine whether, and to what extent, neural network based simulations of subitization and counting have a modular structure as proposed by other connectionist modellers. This review will help in the specification of a modular neural network system for subitization and counting (sections 3 and 4).

2.1 Simulating Subitization

2.1.1 Learning Internal Representation

In their major paper describing the utility of error back-propagation networks Rumelhart et al (1986) solved a number of problems to prove the efficacy of the multi-layer back-propagation network. A number of these examples were problems that related to a network's ability to learn how to quantify, and are listed in Table 2. The emphasis here is on the ability to *learn* how to quantify, something that we have highlighted as being important in quantification skills, especially counting. For subitization, our interest in whether this ability can be learnt stems from the argument that (possibly) innate skills develop through a process of self-organisation, something that can be compared with learning algorithms in ANNs.

Table 2: Internal representations learnt by a multi-layer perceptron (Rumelhart et al, 1986). Network topology is indicated by input dimension and number of units in each layer. For example 'N-N-1' indicates a network with an input dimension of N units, a hidden layer consisting of N units, and an output layer consisting of 1 unit.

Problem	Architecture Notes	Training / Performance
Parity	N-N-1 *N*-hidden units to solve parity problems with patterns of length *N*.	2825 presentations required for recognising 16 patterns comprising 0s and 1s.
Encoding	N-$\log_2$N-N Network for encoding an *N*-bit pattern onto $\log_2 N$ pattern (hidden units) and then producing an output based on the encoding.	Distributed to Local Representation: *m* input patterns mapped onto 2*m* output patterns.
Symmetry	N-2-1 Problem can usually solved by a *2*-hidden unit network.	1208 presentations of a six bit pattern. The trained network has *symmetric distribution* of weights between input and hidden units.
Binary Addition	2N-N-(N+1) Output patterns represent sum of two two-bit number inputs.	3020 presentations of input patterns (e.g. 00+00, 11+11) helped to learn the addition of 16 input patterns on a 4x3x3 network.

In the solution of the parity and encoding problems, the emphasis was also on the position invariance of the input units, which meant that the system learnt to generate a parity signal or an encoding irrespective of whichever input units were on or off.

This invariance of the quantification process is fundamental to the nature of the task: quantification of a set of objects is independent of their physical attributes of location and dimension. Rumelhart et al have demonstrated this point by suggesting that this invariance is somehow *internalised* during the training of their networks – and in their case it was through the adjustment of the weights in the hidden units by the backpropagation algorithm. This is remarkably similar to the *accumulator mechanism* suggested by Gallistel and Gelman (1992) – but more of this later. This invariance is demonstrated further by Rumelhart et al when they turn 'to a more geometric problem – that of discriminating between a *T* and a *C* – independent of translation and rotation' (Rumelhart et al 1986:348). This problem again has the same essence – similar patterns require greater discrimination, as their solutions are radically different.

2.1.2 A Connectionist Approach to Children's Seriation

Mareschal and Shultz (1999) demonstrated a connectionist model of seriation (sorting) that explored how neural networks could be used to simulate the development of psychological abilities in children. Seriation requires an understanding of magnitude and order, and can therefore be linked to an understanding of number. According to Piaget (1952), the development of seriation passes through four stages, which can be identified through the sorting moves made by children and the final configuration of the sorted items. Mareschal and Shultz also summarise the six seriation phenomena that any model must exhibit: periods of constant stage-like behaviour, correct ordering of the four seriation stages, transition between successive stages, better performance with increasing size differences, variation in emergent strategies and gradual stage transitions.

To demonstrate development in a neural network simulation, Mareschal and Shultz used two cascade-correlation networks (Fahlman and Lebiere, 1990), one trained on the ‘which’ task and the other on the ‘where’ task. The input to both cascade-correlation networks was formed by a six-dimensional vector consisting of a rank value for each of the six ‘sticks’ that were to be sorted, with the order detailing the current positioning of the ‘sticks’, the rank equivalent to stick length. The output of both networks was also a six-dimensional vector; the ‘which’ network’s output was used to indicate which ‘stick’ was to be moved by activating just one of the six outputs; the ‘where’ network’s output identified the ‘sticks’ new position.

Cascade-correlation uses an initial network topology without any hidden units, adding hidden units algorithmically if learning does not improve by a given amount within a certain number of epochs of training. Mareschal and Shultz argue that the algorithmic addition of hidden units in the cascade-correlation network during training can be seen as development, more so than connection weight adjustments, even though the training algorithm operates in batch mode differing somewhat from the perceived learning in children.

After testing their network, the authors determined that a bias in the training set was required to ensure that all four stages of development in seriation were simulated. They determined that a higher proportion of less disordered data, as suggested from child psychological data, was required. This was achieved by randomly selecting 50 (out of a possible 720) ‘stick’ configuration inputs that had a sum-of-squares distance from the fully ordered set of less than or equal to 20, and a further 50 with a sum-of-squares distance greater than 20. Matched with each selected input pattern were the target outputs formed as two vectors for ‘stick’ identification and move (complete sets of moves were not used as it was suggested that children learn to sort by witnessing only incomplete sequences). Training proceeded with the selected patterns separately for each network until they were determined to have achieved the fourth developmental stage of seriation by producing four consecutive epochs of output with the required criteria.

The first three experiments performed explored the learning capabilities of the networks with differing ranks (‘stick’ lengths). For each experiment 20 pairs of the ‘which’ and ‘where’ networks were trained, and the stages of development that they achieved and the number of epochs required, recorded. Disappointing results lead to the modification of the training set to include an additional 20 randomly selected input-output pairs drawn from the 24 possible three element series. Of the 20 networks that were subsequently trained, all achieved the desired fourth stage of development.

With these experiments, Mareschal and Shultz demonstrated that a connectionist model could demonstrate the four stages of development in seriation, with the stages and the six associated seriation phenomena emerging as a consequence of learning alone. Furthermore, they suggested that the failures observed with children performing seriation tasks can be attributed to their inability to sort less and less disordered sets, as with the experiments they performed subsequently, exploring the biasing of the input data.

2.1.3 SUBIT-PDP: ‘Subitizing Phenomenon as an Emergent Property of the Human Cognitive Architecture’

Peterson and Simon (2000) have compared and contrasted the performance of a typical rule based simulation of subitization with that of a feedforward, fully connected, three layer neural network that *learnt* to subitize. The rule-based simulation was based on John Anderson’s *ACT-R* system (Anderson, 1993) and incorporates the use of ‘counting’, using a number system, and ‘recognition’, based on the strength of the ‘trace’ left in the memory proportional to the frequency of the stimulus (Anderson, 1993). The rule-based system has a conflict resolution strategy that mediates between the use of the two procedures, a strategy that can be compared within neural computing to a gating network. This type of gated architecture has been implemented by Ahmad and Bale (2001).

Peterson and Simon’s SUBIT-PDP system uses the backpropagation algorithm, investigating the effect of the number of nodes in the hidden layer of their network and its effect on the system’s ability to subitize. The network comprises 16 input and 6 output units and the number of hidden units ranged from 3 to 5. SUBIT-PDP was trained and tested for its ability to subitize up to ‘6’. The test results were positive in that SUBIT-PDP can subitize up to ‘4’, a result which corroborates with observations on humans discussed above. However, the performance of the system depends on the number of hidden nodes and on the maximum numerosity the system has been trained to subitize. The performance of SUBIT-PDP suggested that ‘subitizing emerges through experience in this domain rather than being the result of a hardwired structural limit on the representational capacities of the architecture [SUBIT-R]’ (Peterson and Simon, 2000:102).

For SUBIT-PDP, the numerosity of a set of objects, irrespective of their size and location, is represented on a ‘hypothetical 4 by 4 grid of locations’. The input vector comprises 16 elements, each corresponding to a cell on the 4 by 4 grid: the binary digits, 1 and 0, indicate the presence or absence of an object. The desired output is represented by a 6 element vector to represent numbers 1 to 6. For example, a group of 5 objects can be represented as ‘0100110000010010’. The desired vector for ‘5’ may be represented by a sequence of binary digits: for example, ‘000010’. In this representation scheme, there are only 16 patterns for numerosity ‘1’, 120 for ‘2’, 560 patterns for ‘3’, 1820 for ‘4’, 4368 for ‘5’ and 8008 for the numerosity ‘6’.

SUBIT-PDP was trained on 50000 random patterns and was tested on 25 for each of the six numerosities – making a total of 150 patterns that were presented to the network. At the onset of training all connection weights (input-to-hidden units and hidden-to-output units) are given a random value. Table 3 shows how the learning proceeds for the six numerosities for a 16-3-6 network during a 15000 cycle training regimen observed at every 3000 cycles. The *learning threshold*, that is the minimum activation level of the output unit above which the network is deemed to have learnt a given numerosity, was set by the authors at 0.75. The table shows that whilst the system had no problem in learning the numerosities ‘1’, ‘2’, ‘4’ and ‘6’, there were problems with ‘3’, learnt only after 9000 cycles of training, and the system could not learn ‘5’ in the 15000 cycles.

Table 3: Subitization learning in a 16-3-6 network. If the activation level was above 0.75, then the network is deemed to have successfully learnt and recognised the pattern and this is denoted by a √. Otherwise learning is deemed to have been unsuccessful and this is denoted by an ×. Note the early 'learning' of the largest numerosity '6'. This persists whenever the numerosity of the training set is increased – the so-called *end effect*.

Numerosity	**Activation After Number of Cycles**				
	3000	**6000**	**9000**	**12000**	**15000**
'1'	√	√	√	√	√
'2'	×	√	√	√	√
'3'	×	×	√	√	√
'4'	×	√	√	√	√
'5'	×	×	×	×	×
'6'	×	√	√	√	√

Peterson and Simon explain this apparent anomaly by arguing that the numerosity of '6' is merely reflecting the 'end effect'; note that '6' is the highest possible numerosity for the simulation discussed above and as such the system needs to distinguish '5' from both the numerosities '4' and '6', but '6' needs to be distinguished from '5' only. Such an observation is in line with the experiments on human subitization where participants appear to have learned the higher numerosities faster than other numerosities in a given counting range (Mandler and Shebo, 1982). They argued that the end effect could be rectified by increasing the number of hidden units in the SUBIT-PDP network, which they demonstrated by further experiments. Here, increasing the number of hidden units seems to merely expedite the learning of all other numerosities, with training taking a larger number of cycles to complete. The results show how the end effect shifts to numerosity '7', which SUBIT-PDP fails to learn in its modified form, despite having a significantly larger number of hidden units. However, rather than confirming their argument, the experiments seem to suggest that the subitization limit is not only related to the number of hidden units used in the neural network, but also to the highest numerosity present in the training set.

The desired vector used by Peterson and Simon to train the network is also quite interesting and relates well to the observations of Gallistel and Gelman's speculation that numbers are represented through the so-called *accumulator mechanism*. Recall the representation used in SUBIT-PDP. The numerosity of one object or a collection of objects is represented by a two component vector. The first component helps to represent the physical location of the object or objects in the hypothetical grid. The second component emphasises the numerosity by a unique code: '1' by '100000', '2' by '010000', '3' by '001000', '4' by '000100', '5' by '000010' and '6' by '000001'. The unique coding is reminiscent of an accumulator filling up (Gallistel and Gelman 1978, 1992). This kind of mechanism ensures that smaller numerosities are encoded more distinctly than higher numerosities.

2.1.4 A Neuronal Model of Elementary Numerical Abilities

Dehaene and Changeux (1993) used an understanding of numerical psychology in order to construct a series of networks that could convert a visual scene input into an internal, abstract representation of numerosity. Their goal in constructing the model was to concentrate upon observed subitization characteristics in order to provide feedback to the understanding of how subitization and internal representation operate within humans and animals.

They used a series of four networks: visual input clusters, an object location and normalisation network, summation clusters and a topographic set of numerosity clusters. Within these networks no learning was used, rather different sets of parameters were chosen at network initialisation. Testing proceeded with 2500 sets of up to 5 objects (500 sets for each number), presented at random locations and with random size on the visual scene. The topographic numerosity clusters resulted in numerosities ordered into a number line, demonstrating both Fechner's law and the distance effect.

Dehaene and Changeux concluded that this representation provided evidence for subitization as an immediate apprehension of numerosity, and not as a process of preverbal counting, as suggested by Gelman and Gallistel (1978). Furthermore, this apprehension was achieved without resorting to the recognition of canonical visual patterns as has been suggested as a suitable mechanism for subitization (Mandler and Shebo, 1982). The limit of 5 objects for subitization was attributed to both the representation of numerosity internally and accuracy of the visual normalisation, leading to the conclusion that the limit may vary between both individuals and species.

2.2 Simulating Counting

2.2.1 Counting Chimes

Connectionist models that have attempted to imitate the numerical competence of counting have focused on either responding to individual items (Amit 1988, 1989, Hoekstra 1992), or the acquisition of the number word sequence (Ma and Hirai 1989).

Daniel Amit's chime-counting modular network has demonstrated how to represent the abstract concept of *number*, as well as that of the counting task, in a neural computing system employing a recurrent network. There are two modules in Amit's network: the first converts an input chime into a *proto-chime*; the second uses the proto-chime to update the state of the network (cf. Dehaene's accumulator) and produces the tally, after detecting a predefined period of silence (no chimes).

The presence of the initial pre-processing of a chime into a proto-chime demonstrates how such a system can be applied to the counting of any type of object or event by abstraction. For Amit, this 'universal counting network' (1989:243), provides a black-box counting mechanism capable of discriminating between different counting sequences and recognising objects within a sequence in a robust way.

2.2.2 Ma and Hirai's Heteroassociative Network

Ma and Hirai (1989) demonstrated how the development of the number word sequence in children could be simulated. They used the combination of a heteroassociative network and a recurrent inhibitory network to simulate the production of the number word sequence as observed from children (Fuson et al, 1982), including stable conventional, stable unconventional and unstable elements. In addition, they demonstrated how learning associations for lower numbers ('4', '5', '6', '7') could influence the production of higher sequences of numbers ('14', '15', '16', '17') and lower incidence of irregular numbers during learning ('15').

2.2.3 A Recurrent Neural Network that Learns to Count

Rodriguez et al (1999) concentrated upon teaching a recurrent neural network to understand a *deterministic context free language* (DCFL). In this case, the language consisted of a string of ‘a’s followed by the same number of ‘b’s, with each letter presented to the network individually. The task of the network was to predict both the next letter when presented with a ‘b’ and when the string of ‘b’s would finish. In this way, the network was taught to count the number of ‘a’s presented, in order to predict the number of ‘b’s. This was achieved with a *backpropagation through time* network trained on sequences consisting of up to 11 ‘a’s (and hence followed by 11 ‘b’s). The network was tested with successively longer strings until it failed to correctly predict the required number, and hence demonstrating how capable it was of generalising. Over 50 trials, they found that 8 networks successfully learnt to predict the correct number of ‘b’s, with one of the networks capable of generalising up to 25. This sort of counting network demonstrates how Gelman and Gallistel’s (1978) concept of subitizing as a form of preverbal counting may be implemented, since counting is not based upon a number word sequence, but only on an abstract understanding of the number of objects presented sequentially.

3 SSUBSYST: A Neural Simulation of Subitization

The presence of subitization, or more accurately the presence of the subitization limit, suggests that ‘many animal taxa […] have a natural ability to discriminate numerosity’ (Brannon and Terrace 1998:747). Peterson and Simon (2000) successfully showed how a neural network could be tutored to learn subitization, in contrast to Dehaene and Changeux’s (1993) hardwiring. From their simulations it is difficult to argue that subitization is a natural ability, in light of the success of the tutoring, despite somewhat intuitive evidence that infants or primates do not have access to and/or understand a tutor in their immediate environment. This suggests that subitization is environmentally inspired.

Dehaene’s latest writing suggests that the ‘human mind starts in life with a rich knowledge about objects, colours, numbers, faces and language’ (2000:42) and that the brain has an ‘internalised representation of numerical quantities’ (Dehaene 2000:43). Ontogenesis plays a key role in the development and maturation of the animal brain; could ontogenesis lead to internalised representations of numerical quantities? We explore this possibility through the use of unsupervised, self-organising neural networks in a multi-net system for simulating subitization. Our system is essentially a sequential multi-net system with two major subsystems: first, a magnitude representation based on Kohonen’s SOM (1997) bi-directionally connected to a verbal SOM; a Hebbian link is used to link the two SOMs. Second, a mapping subsystem which processes a visual scene, comprising objects to be subitized, to map the information in a scale and translation invariant manner onto the magnitude representation SOM.

The sequential multi-net system SSUBSYST (*S*urrey *Sub*itization *Syst*em) was trained to subitize up to five. The performance of the system during testing suggests that it performs just as well as SUBIT-PDP (Peterson and Simon’s system). The performance for lower numerosities is very similar for both the systems; both systems show the idiosyncratic edge effect. The key differences between the two systems are (a) that the sequential system has been trained using unsupervised learning algorithms and (b) the edge effect in our system can be partially attributed to the architecture design of the SOMs in general.

3.1 SSUBSYST Architecture

The architecture comprises three major modules: one module, comprising two networks, for mapping the visual scene onto the magnitude representation network; a second module, comprising one network, for representing the numerosity of objects as magnitudes; and a final module to translate the magnitude representation into an output (see Figure 1 and Table 4).

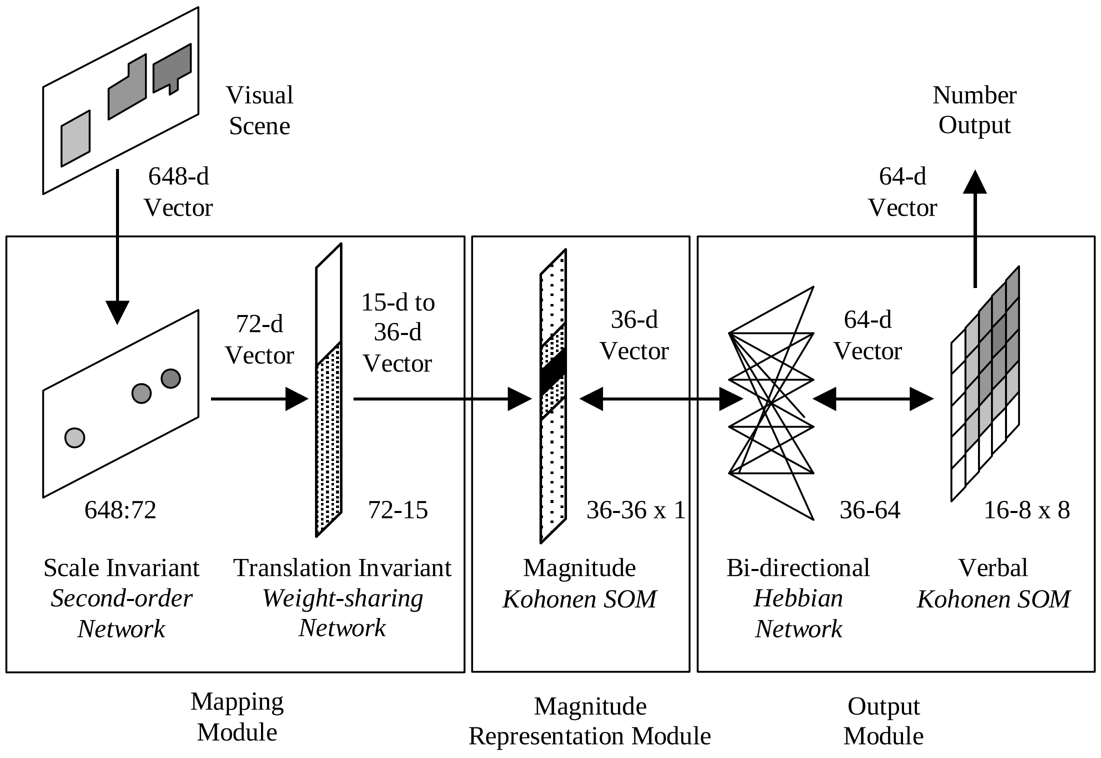


Figure 1: Architecture of SSUBSYST, comprising a mapping, magnitude representation and output module. Constituent network types are shown with relevant vector and network dimensions.

Table 4: Network topology is indicated by input dimension and number of units in each layer. For example '648-72' indicates a network with an input dimension of 648 elements with a single (output) layer consisting of 72 units. For the Kohonen SOM the output layer is represented as a two-dimensional grid of units, for example '36 x 1' indicates a map of 36 by 1 units.

Task	Network	Topology	Input
Scale invariance	Second order	648-72	Visual scene consisting of a 36 by 18 grid. Each 3 by 3 section of the grid represents an object.
Translation invariance	Weight-sharing	72-15	Scale invariant visual scene consisting of a 12 by 6 grid. Each element within the grid represents an object.
Magnitude	SOM	36-36 x 1	Scale and translation invariant visual scene represented as a 36 dimensional vector. The 15-dimensional output from the visual scene is padded to 36 dimensions.
Bi-directional linkage	Hebbian	36:64	For magnitude to verbal mapping, input is the activation of the Magnitude map as a 36 dimensional vector. For verbal to magnitude mapping, input is the activation of the Verbal map as a 64 dimensional vector.
Verbal	SOM	16:8 x 8	Phonetic representation of 22 numbers as a 16 dimensional vector.

3.1.1 Mapping Module

The visual scene consists of a 36 by 18 node 'retina', divided into receptive fields of size 3 by 3, on which several objects can be represented simultaneously. Consider the example of three objects of random shapes, sizes and positions spatially distributed across a portion of the retina (as in Figure 1), albeit restricted to a single row. The mapping module consists of networks for representing the objects independent of their sizes and for representing the objects independently of their positions.

A second-order network receives input taken from the visual scene (Giles & Maxwell 1987). The network consists of groups of nodes each responsible for a 3 by 3 grid of elements. Each node ignores the scale of the object detailed in the 3 by 3 grid by providing a single activation if any two of the elements represent an object. For simplicity it is ensured that a single and entire object is positioned within a receptive field. There is much literature on the subject of visual scene interpretation (Humphreys 1998) and complex connectionist models have been proposed, such as the *Neocognitron* (Fukushima and Miyake 1982). We employ second-order neural networks to execute the subtask of responding to the presence of objects, with levels of activation that are independent of the sizes of the objects or their positions within the receptive fields. A weight-sharing network is then used to map from a spatial arrangement onto a representation that is independent of the positions of the objects across the entire retina. Since the visual scene is not the focus of this study, we use a simple mapping module, capable of distinguishing between the 'what' and 'where' information of simple objects only.

3.1.2 Magnitude Representation Module

The key component of the numerosity detection system is a mechanism for representing small numerosities as magnitudes along a number line. For this, a self-organising neural network, which develops spatially ordered feature extractors, is employed. The module takes an accumulator representation as its input and learns to represent and organise these patterns of activity according to the distance effect and Fechner's law for numbers. Whilst Anderson (1995) has shown that nodes in a two-dimensional Kohonen SOM are capable of representing magnitudes of number, we demonstrate that a self-organising map can represent number magnitude in accordance with the distance effect. This could be modelled as a spatial arrangement in which adjacent regions along a single row of nodes represent consecutive magnitudes.

Our model concentrates on the abstract representation of magnitude on a number line and hence we chose to use a one-dimensional SOM with 15 nodes, with 15-dimensional input. Table 5 shows a set of idealised training patterns where, instead of a shifting pattern of activations as in McCloskey and Lindemann's (1992) and Dallaway's (1994) models, a cumulative pattern of activation is used. Here, because of the way in which the SOM algorithm organises patterns, each pattern must contain some similarity in order for each to be distinguished in relation to all others. For example, the pattern for two (first six dimensions active) contains within it the pattern for one (first three dimensions active).

Table 5: Training vectors used in the Kohonen SOM magnitude representation simulation.

Numerosity	Input Vector
One	[1,1,1,0,0,0,0,0,0,0,0,0,0,0,0]
Two	[1,1,1,1,1,1,0,0,0,0,0,0,0,0,0]
Three	[1,1,1,1,1,1,1,1,1,0,0,0,0,0,0]
Four	[1,1,1,1,1,1,1,1,1,1,1,1,0,0,0]
Five	[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]

3.1.3 Output Module

Following on from the magnitude representation of number, output is required as a response formed as a verbal representation. To achieve this a verbal representation of number is used to describe the magnitude of the numerosity detected in the visual scene connected to the magnitude representing SOM by an intermediate network (as suggested by Gallistel and Gelman 1992 and Dehaene 1992). The learning of bi-directional mappings was modelled by a neural network which linked the SOM representing number as magnitudes, to a further SOM which represented number verbally.

It has been suggested that neural network architectures can be used to generate a topographic map from a pre-synaptic two-dimensional array in a post-synaptic two-dimensional array (see, for example, Willshaw and Malsburg 1976 and Amari 1980). The neurons in the pre-synaptic layer (input) layer are connected to the post-synaptic (output) layer through the use of the Hebbian rule. Our architecture is based upon connecting a Hebbian network between the magnitude representing SOM and the verbal representing SOM; this tripartite network, two SOMs connected with a

Hebbian link, was originally used by Abidi and Ahmad (1997) in their simulation of language development in children. Here the two SOMs can be compared to Willshaw and Malsburg's pre- and post-synaptic sheets.

Hebbian links strengthen the connections linking the most highly activated regions in each SOM. The effect during the training process is the gradual establishment, over time, of associations between magnitude and verbal representations of number. Since Hebbian connections are bi-directional the verbal representation can also be transformed into a magnitude representation, demonstrating how elements of Dehaene's (1992) triple-code model may be constructed. The key element of this multi-net arrangement is that the associations between the different representations is unsupervised and based upon positive feedback from patterns of activation. Therefore a feature map has been used to map an abstract number representation into a symbolic verbal output, as well as being used to form the representations themselves.

3.2 SSUBSYST Training

The various constituents of SSUBSYST were trained individually.

3.2.1 Scale Invariant Network

The scale invariant network consists of a second-order network that also employs weight sharing to duplicate processing over the 3 by 3 receptive fields (Giles & Maxwell 1987). The input is the visual scene represented as a 648-dimensional vector. Each of the 72 nodes within the network is trained to recognise the presence or absence of an object within a 3 by 3 receptive field. The network uses second-order techniques in order to achieve scale invariance. This means that the inputs to each node are first multiplied together before being weighted and then passed through the activation function, which in this case was a hard threshold (or Heaviside) function. As the operation of each node is the same for all of the 72 grouped inputs only one node need be trained. Consequently, a single node was trained using the Hebbian learning rule extended for second-order networks described by Giles & Maxwell (1987) on a set of 8 example inputs representing all possible *gradients* within the grid (for example, objects lying horizontally, vertically or diagonally). Training proceeded with a learning rate of 0.5 for 20 epochs and the resultant node replicated to cover the 72 required grids and therefore producing a 72-dimensional output representing a grid of 12 by 6.

3.2.2 Translation Invariant Network

The translation invariant network consists of a single layer of 15 nodes. Each of the nodes is connected to all of the inputs, which are formed by a scale invariant representation of objects in a grid of 12 by 6. A weighted summation is calculated using all of the inputs and the weights associated with a node, which is then passed through a hard threshold (or Heaviside) activation function. Translational invariance is obtained by training the weights to require a higher input activation to result in an increased number of nodes activated in the output. This is achieved through the sharing of the weights of one of the nodes with all others within the network, essentially only requiring one node to be trained. A selection of training patterns was used to represent a random set of objects within the visual scene and training proceeded with only one pass of the training data required for one of the nodes to learn the required representation. A Hebbian learning rule was used, as with the scale invariant network, with learning rate set to 1.0.

3.2.3 Magnitude Representation Network

The magnitude representing network is formed by a Kohonen SOM which obtains input from the translation invariant expert. The nodes within the SOM were arranged in a 36 by 1 grid representing a number line. Input to the network is a 36-dimensional vector formed as a scale and translation invariant visual scene. Idealised training patterns, (shown in Table 5) representing accumulated activity over the visual scene, act as the inputs to the map. Hence, the sum of activity over the first layer is comparable to a numerosity of between one and five, which the network learns to represent in the second layer.

Initially, the weight vectors are assigned random values so that the competitive nodes that are most closely associated with each input pattern are randomly arranged across the map, reflecting the fact that the map has no knowledge prior to training (as shown in Figure 2a).

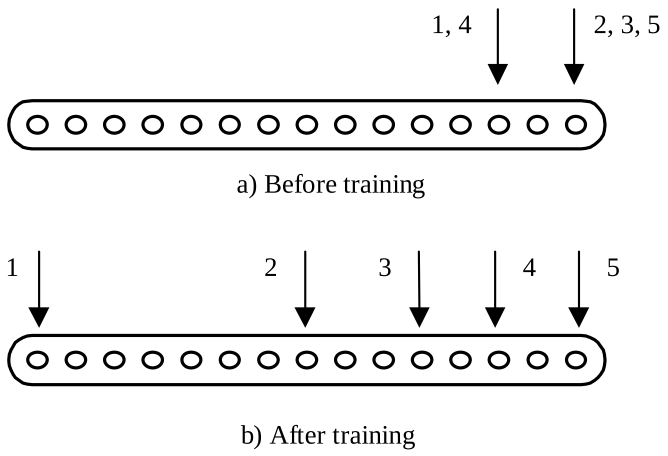


Figure 2: The winning nodes for the representations of five numerosities are randomly positioned a) before training and b) topologically ordered after training of 100 cycles.

The training process causes specific regions of the grid to become associated with particular input patterns by gradually transforming weight vectors towards the input pattern representations. Training proceeded for 150 epochs for each of the training inputs in random order. Initially the learning rate of the algorithm was set to 0.5, and the square neighbourhood size 15. After every 8 epochs, the learning rate was decreased to 85% of its previous value, until a value of less than 0.05 was reached at which point it remained static. The neighbourhood size was also decreased every 8 epochs, with the value reduced by 2 if it was larger than 10, by 1 if it was larger than 5 and by 0.5 if larger than 0. The resulting SOM was found to be capable of recognising the five patterns, each of which represented a numerosity. Figure 2 shows those nodes with the highest activation in response to each input pattern before and after training. Two observations can be made from the trained network:

Distance effect: the winning nodes for each input pattern, which represent numerosities one to five, are ordered topologically after training has taken place; the larger the numerical difference between two numerosities, the further apart on the SOM their representations are positioned (see Figure 2). For example, consider the winning node for an input pattern corresponding to the numerosity of 'one' being situated in closer proximity to the winning node for an input pattern for a numerosity of ‘two’ than that for ‘three’. This is compatible with the distance effect, a feature of the brain where two numbers become easier to distinguish the greater their numerical difference. The effect is displayed in this simulation due to the manner in which the feature map learns.

Fechner’s law: The SOM learns to organise the representations of numerosities in a compressive manner that obeys Fechner’s law; it is an outcome of the training procedure that the locations of the winning nodes are positioned in closer proximity to each other as the numerosities they represent increase.

3.2.4 Verbal Output Module

In order to translate this internal, abstract representation of magnitude into a number response, the next element to be trained was the SOM used to represent the verbal output of the model. This is formed as a map of 8 by 8 nodes, with input taken from a 16-dimensional vector. Training the network involved presenting a set of 22 normalised vectors forming the numbers phonetically from 1 to 22. Each element within the input vector represented a phoneme needed for all 22 numbers, and was populated with a value of 1.0 if it was required within the number, 0.0 otherwise. Training proceeded for 100 epochs for each of the training inputs in random order. Initially the learning rate of the algorithm was set to 0.3, and the square neighbourhood size 6. After every 10 epochs, the learning rate was decreased to 85% of its previous value, until a value of less than 0.05 was reached at which point it remained static. The neighbourhood size was also decreased every 10 epochs, with the value reduced by 1 if it was larger than 0.

3.2.5 Multi-net Training

With these two representations in place, they were connected together by a Hebbian network consisting of a fully connected network with 36 inputs and 64 outputs, matching the 36 magnitude network units and 64 verbal units, respectively. To train the network, inputs were provided to both SOMs, reportedly as encountered by infants (cf. Brown’s or Bloom’s corpus: MacWhinney 1991), representing environmental examples of conceptual and language-dependent information concerning number. For instance, an experience may involve the caretaker of the child pointing towards two toys whilst saying to the child ‘there are two of them’. This was simulated during one training iteration by randomly positioning two objects in the visual scene and allowing the scale and translation invariant networks to provide an input to the magnitude representing SOM. Corresponding to this input, the phonological representation of the number of objects in the scene (the number word) was presented to the verbal SOM. These two SOM activations were then used to train the Hebbian links, which had initial random weight values. A Hebbian learning rule with a learning rate of 0.2 was applied for 50 epochs, strengthening the connections between the most highly activated regions in the maps, with the number of objects and corresponding number word presented in random order.

3.3 SSUBSYST Testing

3.3.1 Testing Strategy

Having individually trained and tested the component networks comprising the mapping module and the magnitude representation network, the SSUBSYST numerosity detection system was constructed by linking the networks together in series. Testing of the numerosity detection system involved presenting novel input patterns by introducing Gaussian random noise (with $\mu = 0$, $\sigma = 0.1$) to the input nodes in each receptive field occupied by an object. In response to this input, the mapping module generated patterns representing accumulated activity over the visual scene. The magnitude representation network then received these noisy versions of the representations on which it had been trained. To suppress noise in the output of the system, the winning neuron was depicted with peak activation surrounded by neurons with decreasing activation. This took the form of the winning node being allocated an activation level of 1, whilst its neighbours received activation in proportion to their distance from the winner, according to a Gaussian distribution.

It was found that presenting a specific number of objects in the visual scene caused an appropriate magnitude representation to be activated, as in Thompson et al's (1970) observations on cat neurons. The activation of the magnitude representation was then used to activate the verbal representation (magnitude to verbal mapping) via the Hebbian connections. The property of bi-directionality in the Hebbian links enables magnitude representations to be retrieved in response to an input of a number word representation; by having as input a verbal representation of number to the relevant SOM, a specific region on the map with a magnitude representation for number is caused to become activated (verbal to magnitude mapping).

3.3.2 Validating SSUBSYST

Since Dehaene and Changeux's (1993) model has a number of similar attributes to SSUBSYST, namely visual scene transformation and topographic numerosity representation, and also provided a viable simulation of the proposed subitizing mechanisms, we compared Dehaene and Changeux's results with those obtained from SSUBSYST.

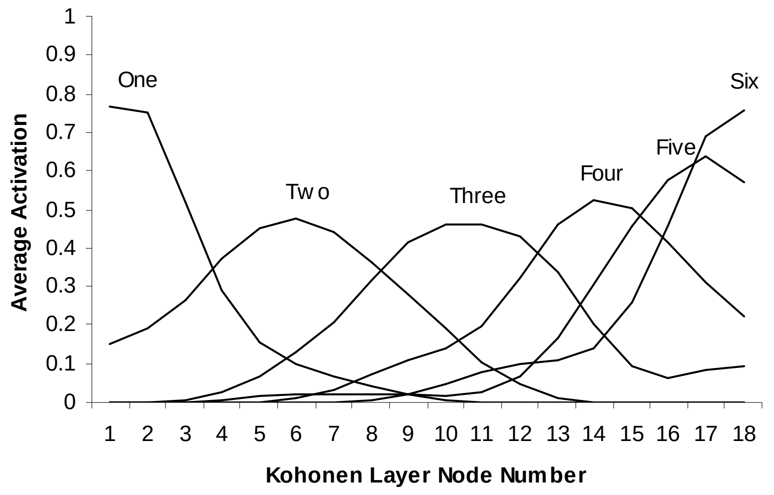
To achieve this, our system was presented with 500 examples of objects of random shapes, sizes and locations across the visual scene for each of the numerosities one to five. To allow comparison of the internal representation of magnitude, the output from the magnitude SOM was recorded, rather than passing activation through to the verbal map. Table 6 shows a representation of activations output by our simulation and those reported by Dehaene and Changeux. Specific, yet overlapping, regions of the output layers of both models respond to the number of objects represented on the retina. As seen from the table, the distance between two regions decreases with an increase in numerosity (cf. Fechner's law). In Dehaene and Changeux's system, the effect of Fechner's law is a result of the representations of the numerosities, along a linear number line, having increasing variability as the magnitudes of the numerosities increase (cf. scalar variability assumption). This can be seen by the increasingly wider curves for larger numerosities. In our system, however, there is little difference in the variability across numerosities. The effect of Fechner's law holds here due to the numerosities being represented along a compressive number line (cf. compressive mapping assumption). Fechner's law states that the perceived intensity of a number stimulus is proportional to the logarithm of the actual intensity, hence the internal representation of number is compressed at higher magnitudes. Looking at the pattern of activation for successive numerosities, we can see that each activated region overlaps more and more as the numerosity increases, hence the higher numerosities seem to be compressed and is logarithmic in nature.

Table 6: Representation of the activities output in response to 1 to 5 objects presented in the visual scene of our system and that of Dehaene and Changeux's (1993). An activity value over 0.75 was used to indicate a cluster response for our system, and a value over 0.4 in Dehaene and Changeux's due to the lower levels of activity for higher cluster numbers.

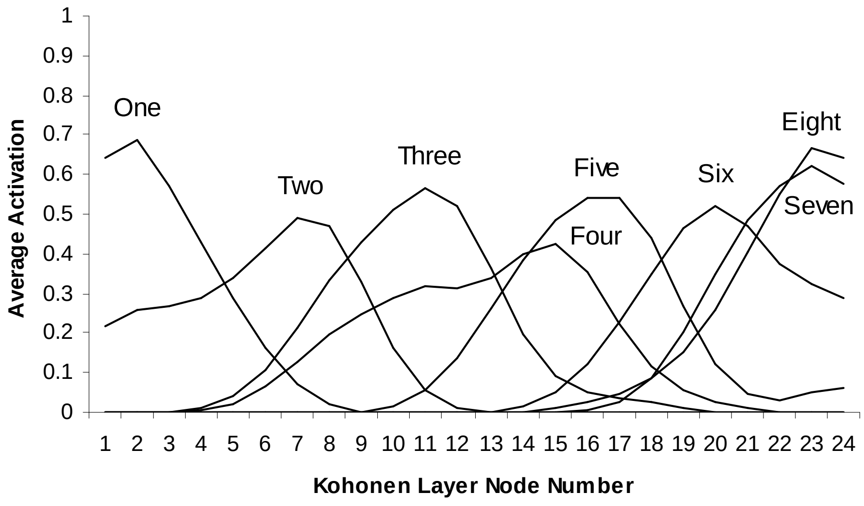
Numerosity	**Clusters**														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Our Results															
'1'	√	√													
'2'							√	√	√						
'3'										√	√	√			
'4'												√	√	√	
'5'														√	√
Dehaene and Changeux's Results															
'1'		√													
'2'					√										
'3'							√	√							
'4'										√					
'5'												√			

In summary, the results of the two systems are similar; in both numerosity detection systems, the representations of numerosities obey the distance effect and Fechner's Law for numbers. The difference is that, in ours, these are side effects that have arisen as a direct consequence of the training procedure employed, whilst these effects were obtained in Dehaene's model through the hard wiring of the connection strengths. For us, the change of weights of the nodes of an interconnected system ensures that it has learnt. The organisation of these patterns is as a result of the way in which the SOM algorithm has organised similar patterns together on the one-dimensional map, looking at similarities in the input data and using the Euclidean distance to cluster like representations.

The comparison of the unsupervised SSUBSYST with Peterson and Simon's supervised network shows interesting similarities and differences. In order to compare the two networks, SSUBSYST was trained to recognise numerosities of up to *six* and then trained again (independently) to recognise numerosities of up to *eight* (see Figure 3a and Figure 3b respectively).



a) Results of training up to six



b) Results of training up to eight

Figure 3: Results of SSUBSYST trained up to the numerosities: a) six, b) eight.

Similarities (cf. Peterson and Simon 2000):

(i) Both networks learn numerosities, numbers '1'-'3' *easily* in their own way. Peterson and Simon's 16-3-6 network needs 6000 cycles of training to recognise '1' and '2' and by the 9000th cycle, the network has learnt '3' as well. If the hidden nodes are increased by one (16-3-6 to 16-4-6) then the three numerosities are learnt in the first 3000 cycles by training. The final activation level for '3' is the same. Much the same is true when Peterson and Simon trained their network for higher numerosities of up to eight using a 36-5-8 network.

SSUBSYST learns numerosities 1-3 in 150 epochs of training (as it does with 4 and 5) and makes a clear distinction between the three in terms of allocating unique (sets of) nodes in the output layer.

(ii) Both networks show an *edge effect* for the lowest and the highest numerosities. For example, numerosities '1' and '6' show the highest activation level in both networks and are learnt quickly by Peterson and Simon's network (within

6000 training cycles) and uniquely by SSUBSYST. If the networks are trained to *subitize* higher numerosities, of up to 8, again both learn the numerosities ‘1’ and ‘8’ quickly.

(iii) Intermediate numerosities, for example, ‘3’ and ‘5’ for networks trained to learn up to ‘6’, and ‘5’ and ‘7’ for networks trained to learn up to ‘8’, are learnt over a larger number of training cycles by Peterson and Simon’s network. In SSUBSYST there is a similar phenomenon observed in that the activation level of nodes that learn intermediate numerosities is lower than that of nodes that recognise ‘1’ and the highest numerosity (either ‘6’ or ‘8’).

Differences (cf. Peterson and Simon 2000):

(i) In one sense one cannot compare the performance of a supervised network with that of an unsupervised network in that the basic learning mechanism is different. Nevertheless, like all backpropagation networks, Peterson and Simon can always improve the performance of their networks by adding yet another hidden node. Such incremental addition may lead to an over-determination of the solution. In unsupervised networks there is no such ‘tweaking’ mechanism available, except perhaps for the neighbourhood size and learning rate.

(ii) SSUBSYST simulates subitization more intuitively; were it not for the paucity of experimental or observational data, a situation which may improve through the efforts of Dehaene and his colleagues, one could argue that SSUBSYST agrees with the observational/experimental data better than Peterson and Simon’s network.

SSUBSYST’s performance simulates subitization in that SSUBSYST’s output conforms to the observations of Fuson, as stated under the rubric of *distance* effect, and that of Fechner, as implied in his eponymous *law*; lower numerosities are discriminated whilst higher numerosities cannot be distinguished easily. The activation level for nodes that ‘win-over’ lower numerosities is typically higher than that of nodes that had ‘won-over’ higher numerosities. This discrimination prevails despite the edge effect.

4 SCOUSYST: A Neural Simulation of Counting

Counting involves the imposition of order on a collection to allow each item within the collection to be tallied one by one. This order is imposed in a stimulus independent manner: whether the stimulus is acoustic, presented for example as chimes emanating from a bell one at a time, or the stimulus is visual, for instance a collection of individuals presented all at once; counting involves keeping track of how many individuals or chimes have been taken into account thus far and to ascertain that the stimulus has ceased.

Counting is also a learnt process: in order to count one has to have some knowledge of a number system, something that is taught. During counting the labels have to be recited such that they match the ascendancy of the numerical sequence and hence there is some association and storage of a number word with an object within the collection. Whilst there may be many different ways in which counting may take place, for example listening for chimes or counting objects in a row, we take a somewhat simplistic approach by assuming we have a visual scene in which objects are presented within a single row. We also assume that there is an indication within the visual scene as to the next object to be counted, much like a pointing finger. With this simplified view, we can determine that the association and storage of a number word within the sequence is dynamic, since an internal representation needs to be kept of the current number word, whereas the identification of the next object is static because our visual scene contains this information, which only needs to be updated.

Within these simplified bounds we have developed a system that uses both sequential and competitive multi-net processing to simulate counting. The system comprises two major sub-systems: the first sub-system transforms a visual scene into a scale invariant output for the second system where the counting takes place; the same scale invariant network used for our subitization system SSUBSYST was used here. The counting sub-system comprises a recurrent backpropagation network for articulating the numerosity of the individuals (counted thus far) in the collection, and a static backpropagation network for the next-object task, operating in competition so that different modules perform word-articulation and next-object pointing independently. The output of the two sub-systems is controlled, or rather gated, by two feedforward networks: one is the number word gate and the other is a next-object gate. The gated output is passed onto a Madaline network, which produces the number output of the collection presented in the visual scene.

The sequential multi-net system, SCOUSYST (*S*urrey *Cou*nting *Syst*em), was initially trained to count up to 5 objects, and then re-trained on larger numbers of objects. The longest row of objects presented to the model was 22 under training, and 29 for testing the model's capacity for generalisation. These numbers correspond closely to the numbers of objects counted by children aged 3½ to 6 in Fuson's (1988) study where the longest rows ranged from 22 to 31 objects. SCOUSYST was trained to count by decomposing the counting task into that of number word update/storage from the next-object pointing task. In this respect, we believe that SCOUSYST is different from other neural counting systems reported in the literature.

4.1 SCOUSYST Architecture

The architecture for SCOUSYST comprises three major modules: one module, comprising one network, for mapping the visual scene onto the counting module; a further module, comprising four networks, for counting; and a final module for outputting a response (see Figure 4 and Table 7).

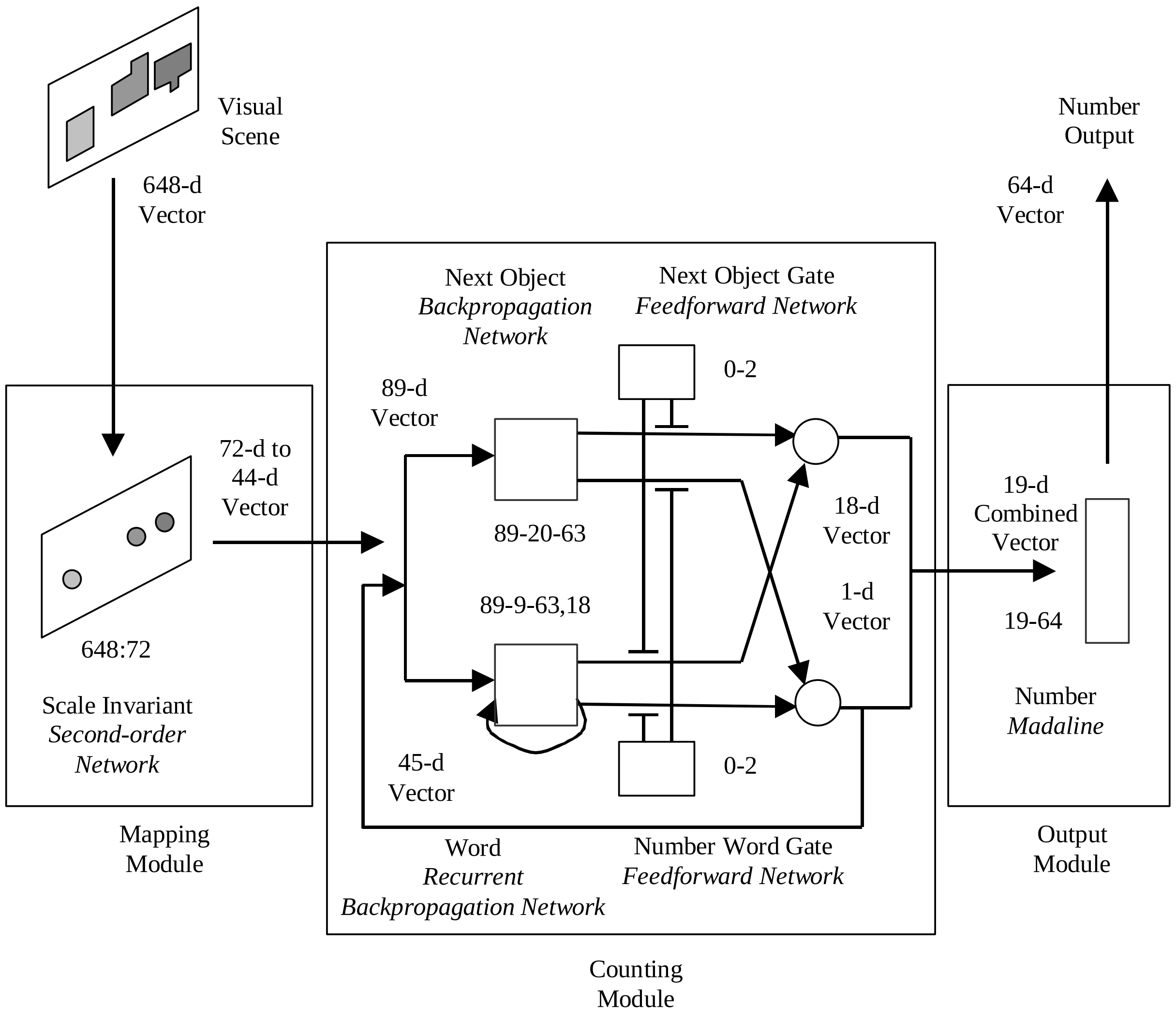


Figure 4: Architecture of SCOUSYST, comprising a mapping, counting and output module. Constituent network types are shown with relevant vector and network dimensions.

Table 7: Details of system employed in modelling verbal counting. Network topology indicated by input dimension, number of units in each layer and number of state units. For example '89-9-63, 18' indicates a network with an input dimension of 89 elements with two layers of 9 and 63 units with 18 state units appended to the first layer of the network.

Task	Network	Topology	Input
Scale invariance	Second order	648-72	Visual scene consisting of a 36 by 18 grid. Each 3 by 3 section of the grid represents an object.
Next object	Backpropagation	89-20-63	Combination of visual scene and 'next-object' output as an 89-dimensional vector. Visual scene is converted from a 72-dimensional vector to a 44-dimensional vector with spaces between objects. Initial 'next-object' feedback has no objects being pointed at.
Number word	Recurrent backpropagation	89-9-63, 18	
Next object gate	Feedforward	0-2	None.
Number word gate	Feedforward	0-2	None.
Number output	Madaline	19-64	Counting network response 19-dimensional vector constructed as an 18-dimensional number word subtask output combined with the 1-dimensional 'no object' subtask output.

4.1.1 Mapping Module

The architecture of the scale invariant sub-system for visual scene analysis in SCOUSYST is identical to the scale invariant element of the subitization system reported above (section 3.1.1). The former does not have a translation invariant sub-system. Since the scale invariant visual scene outputs objects within a receptive field, with no gaps between objects, this was modified before presentation to the counting module to include gaps at appropriate points. The output of the mapping module was further reduced in dimension to match the total number of objects that the counting route was capable of detecting. The remaining input nodes represented the object currently being pointed to, with the activation of the final node representing a 'no point' action. Of these, an input node took an activation value of 0.9 to represent a pointing action and a value of 0.1 otherwise (see example in Table 8).

Table 8: Example inputs to the counting module for three objects, with each object being selected as the next-object until the end. Total input vector dimension is 44 for the object positions, 44 pointing positions and 1 for 'no point'.

Input Vector			Number Word
Objects (44)	**Pointing Position (44)**	**No Point**	**Output**
[**1**,0,**1**,0,**1**,0,0...0,	0.1,0.1,0.1,0.1,0.1,0.1,0.1...0.1	0.1]	-
[**1**,0,**1**,0,**1**,0,0...0,	**0.9**,0.1,0.1,0.1,0.1,0.1,0.1...0.1	0.1]	One
[**1**,0,**1**,0,**1**,0,0...0,	0.1,0.1,**0.9**,0.1,0.1,0.1,0.1...0.1	0.1]	Two
[**1**,0,**1**,0,**1**,0,0...0,	0.1,0.1,0.1,0.1,**0.9**,0.1,0.1...0.1	0.1]	Three
[**1**,0,**1**,0,**1**,0,0...0,	0.1,0.1,0.1,0.1,0.1,0.1,0.1...0.1	**0.9**]	-

4.1.2 Counting Module

The counting system comprises a multi-net architecture formed from three sub-systems: 'word', 'next-object' and a set of decision gates mediating output. The multi-net system is based upon the mixture-of-experts (ME) architecture defined by Jacobs et al (1991). Here, expert networks are trained in-situ with gating networks that learn how to decompose tasks to each of the experts. The ME architecture employs a supervised learning algorithm, but crucially the gating networks use an unsupervised learning algorithm to ensure that the task decomposition is competitive and influenced only by the performance of each individual expert. In this way, experts are selected to perform subtasks and then promoted to perform them better with training.

The counting system employs two expert networks for the 'word' and 'next-object' subtasks, together with two gating networks that mediate production of the 'word' and 'next-object' outputs. Here each of the gates selects an expert network that is optimal for the designated subtask.

4.1.3 Output Module

The output from the counting system forms a combination of 'number word' and 'next-object' representation, including 'no-object' within the response. In order to convert this to a number response, where a number word response is only produced once all objects have been counted, the output was fed into a Madaline network.

The Madaline network, using a Signum activation function, has a single layer of 64 nodes taking input of the 18 possible number word phonemes, together with input representing 'no point'. The output layer of nodes consisted of two subsets; one subset representing a possible verbal response and the other describing the object to which the next pointing action was to be applied. Each of the 18 output nodes in the 'word' subset were associated with either an entire number word or part of a number word as follows ['-teen', '-ty', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten', 'eleven', 'twelve', 'twen-', 'thir-', 'fif-']. The 'next-object' output nodes described the object to which the next pointing action was to be applied, with the final node indicating the end of the counting task. The largest number of objects being presented in the visual scene determined the total number of output nodes in this subset.

4.2 SCOUSYST Training

Both the feedforward and recurrent networks were updated according to the backpropagation algorithm, however the recurrent network utilised state units to provide internal memory (Elman 1990). The weights of the recurrent links connecting the output nodes to the state units were set to values of 1, whilst the weights of the self-recurrent links to the state units were assumed to be 0. A method of teacher forcing for training the recurrent expert was employed, in which the output units send the ideal outputs supplied in the training data set to the state units. We incorporated a further set of recurrent links that connected the output nodes of the 'next-object' subtask with the input visual scene. These links, with pre-set weight values of 1, fed the model's decision of which object to next point at back into the visual scene, and hence updated the input layer for the subsequent time-step.

4.2.1 Scale Invariant Network

On separate runs of SCOUSYST the sizes of the layers of the network were modified according to the size of the problem task. The size of the part of the visual scene for representing objects was set to be twice the size of the maximum number of objects being counted, allowing for spaces to be included between neighbouring objects. Otherwise, training proceeded as for the scale invariant network within SSUBSYST (section 3.2.1).

4.2.2 Counting Module: Training the Gated 'Word' and 'Next-Object' Subsystems

The training data set comprised 50 examples of counting various sized sets of objects, totalling 700 training patterns. Because this training set consists of examples that count through the objects starting at one, there is a bias towards the lower numbers. For instance, for two example visual scenes, one of three objects and one of four objects, the training sets would consist of data counting through 'one', 'two' and 'three' and 'one', 'two', 'three' and 'four', respectively. Thus in this example, the representations for 'one', 'two' and 'three' are presented more often than 'four'. Training proceeded for 400 epochs with a learning rate of 0.95 for the recurrent and feedforward experts, and 0.05 for the gating networks.

The ability of the model, not only to learn but also to decompose the counting task, involved examining the behaviour of the individual experts and the gating networks, in addition to the model as a whole. We discuss each of these below.

4.2.2.1 Learning

SCOUSYST was trained to output a verbal response if a subset of the nodes resembling a number word took a high activation value (greater than 0.6), whilst the remainder had low activation levels. Under a teacher forced training procedure, the model was found to have correctly decomposed the counting task into the two subtasks 96% of the time over 25 trials. For the successful trials, the proportions of correct responses of the model over 400 training epochs for each of the two subtasks were recorded, in addition to the overall counting task itself. The counting task performed only as well as the least successful of its subtasks, which is the 'next-object' subtask at all epochs.

4.2.2.2 Task Decomposition

Throughout the training process, the behaviour of each gating network was studied to confirm that an efficient decomposition of the counting task took place. The two networks are responsible for gating different sets of components' output by the expert networks. The most efficient solution is then for the gate responsible for the patterns describing the 'word' subtask to switch on the output of the recurrent expert and to switch off the output of the feedforward expert and vice versa for the second gate, which is responsible for the 'next-object' patterns. During training it was found that both experts initially favouring the recurrent expert accounted for 36% of the trials. Alternatively, on 12% of the trials, both gates displayed an initial preference for the feedforward expert. On a single trial (4%), the gates chose the most unsuitable combination of experts but they permanently swapped choices after a few epochs. Cases in which the two gates made the most efficient choices from the start of training account for the remaining 48% trials.

4.2.3 Output Network

The produced output indicates the number of items generated by the counting response. The form of this matches the output of the SSUBSYST's verbal SOM (section 3.1.3), namely a 64-dimensional vector with 1-dimension each representing the numbers from 1 to 22. A set of 44 training patterns was used to train the network over 50 epochs with a learning rate of 0.2. The network was trained to output a number word when the 'no point' input was highly activated, with the output corresponding to the input number word. Otherwise, the output represented a 'no output'.

4.3 SCOUSYST Testing

4.3.1 Testing Strategy

Once all networks had been successfully trained, they were connected together to allow testing of SCOUSYST. Following the testing procedure established in testing SSUBSYST (section 3.3.1) visual scenes were presented to the system and activation allowed to propagate through the constituent networks. The result of each time-step of the counting process was recorded, together with the final number response. As used for the subitizing system, random visual scenes were presented and activation allowed to propagate through the constituent networks. The result of each time-step of the counting process was recorded, together with the final number response.

Testing of the counting model involved feeding back the actual output of the recurrent expert network into the state units, rather than the ideal output (which was the method used under the teacher forced training method). In other words, the recurrent expert processed the information it fed back to itself regardless of whether a correct or incorrect number word had been generated on the previous time-step. Another way in which the testing method differed from the training one was that whichever object the model chose to point to next was reflected in the visual scene on the following time-step. This contrasted with the training process in that when the incorrect object was selected, the input on the next time-step was modified to display the object that should have been pointed to.

4.3.2 Comparison of SCOUSYST's Output with Child Development Data

SCOUSYST's output was compared to the observations of counting errors made by children between the ages of 3½ years and 5½ years (Fuson et al. 1982). For example, Table 9 shows two sets of responses provided by one run of the model after 40 epochs, and one after 120 epochs.

Table 9: Production of the number word sequence by the model at stages throughout the training process. Incorrect values in the sequence are shown in bold.

After 40 Epochs	After 120 Epochs
1, 2, 3, 4, 5, 6, 7, 8, **8**	1, 2, 3, 4, 5, 6, 7, 8, **8**
1, 2, 3, 4, 5, 6, 7, 8, 9, **8**	1, 2, 3, 4, 5, 6, 7, 8, 9, **8, 9**
1, 2, 3, 4, **6, 7, 8, 19, 13, 18, 19**	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, **8**
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, **8**	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, **8**
1, 2, 3, 4, **6, 17, 18, 19, 20, 1, 8, 19, 20**	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, **6, 7, 8, 9, 8, 9**
1, 2, 3, 4, 5, 6, **17, 18, 19, 8, 19, 20, 8**	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, **8, 9**

Referring to Fuson et al's categories, we have: a conventional portion; a stable, non-conventional portion; and an unstable, non-conventional portion. Our simulation seems to produce corresponding results:

1. Conventional portion: numbers early in the sequence are generated correctly. For example, after 40 epochs a minimum of the numbers 'one' to 'four' are produced correctly. After 120 epochs this increases to the numbers 'one' to 'eight'.
2. Stable, non-conventional: some numbers are inaccurately repeated consistently out of order. For example, after both 40 and 120 epochs '8' appears in a nearly all sequences at a higher position. The numbers 'seventeen', 'eighteen', 'nineteen' and 'twenty' also exhibit some stability in the 40 epoch sequences.
3. Unstable, non-conventional: somewhat random use of numbers inaccurately, with different numbers used in different sequences. The sequences for 40 epochs alone exhibit these characteristics (for example, 'thirteen'), showing that further cycles of training reduce this occurrence.

The reason that the model is able to consistently and correctly produce the lower end of the number word sequence, and not the remainder, is the imposition of a domain-specific constraint upon the inputs to the model. As discussed, this constraint took the form of a bias in the training data set whereby number words earlier in the sequence occur more frequently than later ones. It has been argued whether certain number words are more frequent in our vocabulary than others (for example, Dehaene and Mehler 1992) but since counting even small sets always includes the number words earlier in the sequence, that is, 'one', 'two' etc., we assume a child being taught to count has an increased exposure to these. SCOUSYST experiences examples of the number word sequence in precisely this manner. By comparing the number word sequences produced by the model at the two stages of training in Table 9, the size of the stable, conventional portion can be seen to increase whilst the other portions decline. Learning of the correct association between neighbouring number words for the higher numbers is a result of further experience with those words.

The second observation by Fuson et al. (1982) concerns the less frequent occurrence of irregular number words in the child's number word output. Similarly, the model was found to have more difficulty in producing an irregular number word such as 'fifteen'. This may be explained by considering the representations of neighbouring number words. Learning to associate regular pairs of number words is aided by past experience of learning other neighbouring number words that share part of their representations. For example, knowing that 'six' precedes 'seven' helps learning that 'seventeen' follows 'sixteen'. Meanwhile, learning that 'five' follows 'four' hinders the task of learning that 'fourteen' is followed by the irregular 'fifteen' rather than 'five-teen'.

In examining pointing errors the aim of the analysis of error production was not only to investigate whether the model gave similar types of errors to those found in children; in addition, we examined whether the *proportions* of those errors were comparable and whether those proportions decreased over time at a similar rate, reflecting the improvement in counting experienced by children. In order to simulate this developmental progression, training was stopped at regular intervals at which snapshots of the model were recorded before training resumed. The snapshots of the model were tested to see whether they corresponded to specific stages in the child's development regarding counting.

To assess the 'next-object' errors made by the model, the 50 examples of rows of objects that acted as input during training were presented to the model again. This was repeated for each snapshot recorded at intervals of 40 training epochs. On each time-step the actual output of the pointing task fell into four categories: firstly, an object selected, represented by a node with an activation level over 0.6 and all other nodes with activation below 0.4; secondly, a ‘no point’ output, with the ‘no point’ node activated over 0.6 and all others below 0.4; thirdly, a ‘best guess’ output describing a category which includes all poorly represented responses. Here an object selected or ‘no point’ output was determined by the node with maximum activation above a threshold of 0.3; and, finally, ‘no output’ indicated by all nodes being activated below 0.3.

If the output on a particular time-step was incorrect, it was classified as falling into one of the following error types: an ‘object skipped’ error occurred if any number of objects were skipped over; a ‘multiple count’ error was recorded whenever the object currently being pointed to was selected again; a ‘no object’ error occurred if the model selected a space between objects; and, a ‘stopped early’ error was considered to take place if objects to the right of the last one being pointed to were not included in the counting procedure. The condition that indicated that the counting task had stopped early was either an incorrect ‘no point’ output or a ‘no output’ (this may be interpreted as a child refusing to continue with the counting procedure). Additionally, a ‘stopped early’ error was recorded whenever the model starting re-counting objects. Often this was accompanied by the number word sequence starting from ‘one’, indicating that the model was attempting to carry out a new counting task. The last error type identified in our model may be interpreted as the end of the current counting task.

According to Fuson (1988) ‘object skipped’ and ‘multiple count’ are the main two point-object error types made by children. Since developmental data is available for each of these error types, they were compared to the proportions of errors generated by the simulation. In Figure 5 the error rates made by the simulation for each problem type are plotted together with Fuson’s data. The number of training epochs corresponding to each of four age ranges, 3½-4, 4-4½, 4½-5 and 5-5½ are chosen to be evenly distanced. Referring to Figure 5, the skipped error rate at 120 epochs can be seen to most closely resemble the data of the first age range 3½-4 and at 200 epochs for the second age range 4-4½. The interval of 80 epochs is then applied in positioning the subsequent children’s data points. By ensuring that the length of training of the network between the points matched is constant, there is an underlying assumption that children are exposed to equal amounts of counting examples between 3½ and 5½ years of age. It can be seen that the proportions of errors for both data sets are fairly similar and both decrease in a comparable manner. The same intervals between epochs were used in plotting the children’s multiple count error rates. Here, the proportions of errors are lower for both data sets and for neither do the rates smoothly decrease with time.

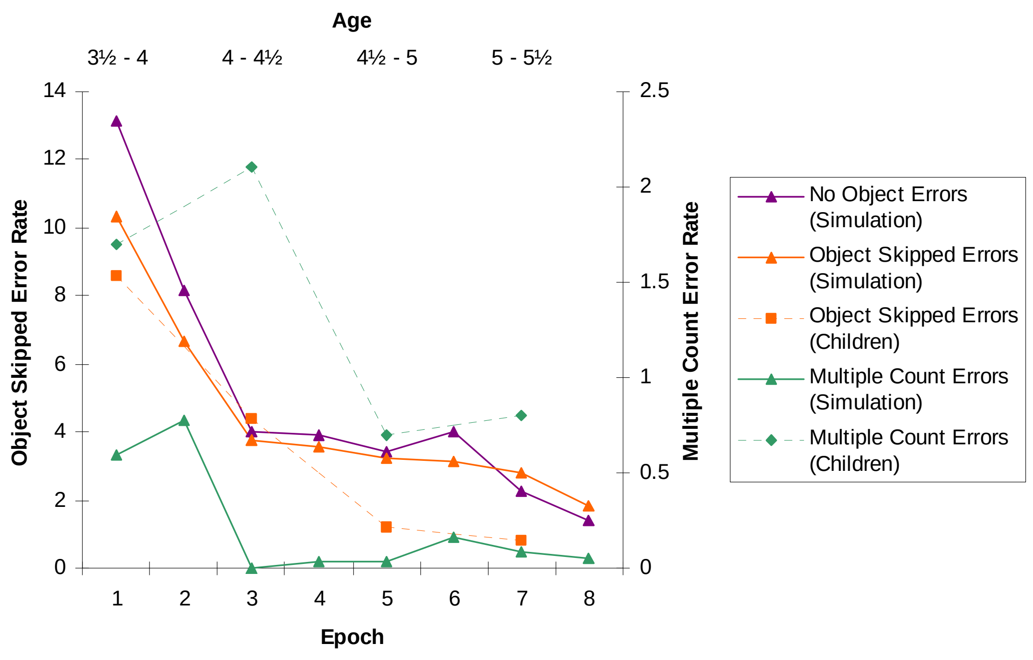


Figure 5: Relationship between proportions of object skipped and multiple count errors by children and by our counting simulation. No object errors are also shown.

Comparison of error types confirms, to some extent, that the model is good for simulating the learning of the next-object subtask. Moreover, the proportions of skipped and multiple count errors found in the model closely correspond to children's developmental data. The network, however, appears to frequently make what can be called a 'no object' error (see Figure 5); according to Fuson (1988) children rarely appear to commit this kind of a mistake (and hence there is no comparative data). This indicates one of the limitations of this model compared to human ability; it reflects the improved ability of a human in distinguishing objects from spaces over that of the model's. Whilst the model lacks knowledge of objects in spatial arrangements, it has been reported that infants are capable of interpreting the physical world in terms of individual entities from an early age (Spelke 1994). This may be accounted for by the simplistic treatment of the visual scene used in constructing this simulation.

4.3.3 Generalisation in SCOUSYST

SCOUSYST was trained with a maximum of 22 individuals in a visual scene. If the system has *learnt* to count, then perhaps it could cope with a larger number of individuals. The system was tested on visual scenes comprising up to 29 objects.

For the 'word' subtask, verbal responses of number words higher than 'twenty-two' had not been encountered during training. Despite this, the counting model attempted to represent number words according to the limited experience it had gained. Representations for number words output by the model during testing were 'one' through to 'nineteen', 'twenty', 'twenty-one', 'twenty-two', 'three', 'twenty-four', 'five', no output, 'twenty', no output and 'nine'. It is not surprising that the first three number words in this list are correct since they formed part of the training set. Although the representation of the next number word, 'twenty-three' was not achieved, the model did output a number word by using its experience in associating 'two' with 'three'. On the following time-step, the number word 'twenty-four' was successfully represented by activation levels over 0.5 in the elements of the vector symbolising the syllables 'twen-', '-ty' and 'four', and by activation values less than 0.5 elsewhere. Even though higher number words were poorly represented by the model, the correct output of 'twenty-four' is indicative of some capacity of the model to generalise.

Simulation of the production of the number word 'twenty-four' was possible, despite failing to produce the number word of 'twenty-three', through the teacher forced method of correcting the actual output of the model from the previous time-step. This simulates prompting a child with a correct number word once he or she has generated an incorrect one, in order to assist in the retrieval of the succeeding word in the sequence. Perhaps like children, the model was able to use experience of associations between neighbouring number words to generate a number word that it had not encountered previously. However, there is a limit for such a model to generate unseen number words. The production of irregular terms in the number word sequence, such as 'thirty', would require explicit teaching, in the form of training on a larger data set. This is perhaps demonstrated for the training of the numbers 'eleven' and 'twelve', as they do not form a 'teen' value (see Butterworth 1999, for a discussion on a child's learning of the English number words).

A limitation of the model in pointing to objects at unseen locations was also identified. The response of the 'next-object' subtask was to point to each object correctly in turn when the locations of objects were exemplified during training. Under a localist representation scheme, individual positions of objects are denoted by particular elements of the input vector. If the training data set lacks an example of an object being located in, say, the 14th position, then the corresponding input node would not have experienced an update in its connection weights during the training process. Although this may be solved by ensuring a complete set of training patterns, the network would still not be able to generalise in pointing to a larger number of objects than the maximum sized set presented in training. Two ways in which this might be overcome concern: first, the weight-sharing of connections linking nodes which were involved in the learning process with nodes whose weights had not been; and, second, the use of explicit rules which, McClelland (1995) has proposed, might combine with implicit strategies to direct children's behaviour.

The model presented here is only a first step in attempting to model the co-ordination of the two subtasks involved in counting. Two limitations faced by this model are that firstly, the work deals with one of the less complicated forms of counting whereby the objects are immovable and positioned only in a row and secondly, both subtasks are assumed to be learnt simultaneously whereas children are exposed to, and can recite, the number word sequence before applying it in a counting procedure. However, there is no reason why prior domain knowledge regarding either of the subtasks cannot be incorporated into the relevant expert network in a mixture-of-experts model, for example, through the initial set of weights.

5 Conclusions

In this paper we have reported on two manifestations of numerosity, subitizing and counting. We have reviewed past simulations of these and related abilities, concluding that the majority use either hard wiring of connections or supervised learning techniques. As a consequence, we examined the role of unsupervised learning as a way of modelling these two abilities and presented: a) a collaborative neural network for subitizing, and b) a collaborative neural network for counting.

Subitization is regarded by some as a form of preverbal counting and by others as an innate ‘number sense’; the fact that other animals appear to subitize makes the phenomenon interesting for brain sciences in general. From a multi-net perspective, subitizing appears to involve collaboration between modules, or specialised areas in the brain, that perform visual object recognition, magnitude representation and sound or gesture generation, implying subitization of the number (of objects).

SSUBSYST, a modular unsupervised neural network system, performs as well as, and marginally better than, the single network neural computing system developed by Peterson and Simon (2000): SUBIT-PDP. It also compares well to Dehaene and Changeux’s (1993) model, in which the origin of the modules is clouded. For example, the SUBIT-PDP system shows that the numerosities ‘1’ to ‘3’ are learnt easily. Furthermore, both SSUBSYST and SUBIT-PDP exhibit an *edge effect*, having difficulty learning intermediate numerosities in the range above ‘3’. This comparison shows that our unsupervised network appears to learn as well as or better than Peterson and Simon’s. However, the similarity between the occurrences of the edge effect are superficial; Peterson and Simon attempt to explain that the edge effect is dependent upon the input data, with larger numerosities being more infrequent in subitization than smaller numerosities. In contrast, because the SSUBSYST network uses the unsupervised learning paradigm, the edge effect can be explained because of idiosyncrasies in Kohonen’s SOM learning algorithm.

Peterson and Simon also appear to improve their network’s performance by adding more hidden layer elements, affecting the highest learnable numerosity and the range of intermediate values that suffer from poor learning. Whereas the highest learnable numerosity in SSUBSYST can be increased by making the size of the Kohonen map larger, the overall performance is still subject to the self-organisation of the different magnitudes into a number line, with higher numerosities being closer and closer together, comparing well the observations of child numerosity, namely Fechner’s law and the distance effect. Because of this self-organisation phenomenon the use of Kohonen maps, or more particularly unsupervised learning, makes SSUBSYST more plausible than the supervised architecture used by Peterson and Simon. Our argument of plausibility, how ever weak, relies on the fact that it is nearly impossible to *teach* a neonate the difference between quantities.

Above all, for us it is the modular nature of our network that provides a more plausible simulation of cognitive abilities, and perhaps contributes to the more ambitious projects of Dehaene, who focuses on “charting meaning in the human brain [which] requires [..] discovering whether and how some of these features [of a number system] have been extracted in the course of cerebral evolution and have been internalized in the brains of infants and animals” (2000:42).

It is worth noting here that our agreement with Dehaene (2000) on subitization is gratifying in that his triple-code model comprises modules for analogue magnitude, visual Arabic, and auditory verbal processing. Subitization is understood to be performed within the analogue magnitude module, in which an internal representation of numerosity is thought to reside, as we have simulated within SSUBSYST. In contrast, the rote learning of counting is associated with the auditory verbal module, which we simulate with SCOUSYST.

Counting can be viewed as an inherently recursive task; as one counts a set of objects, that is, makes a progression along a line, one has to refer back to the last object one has counted. Also, if one has to articulate the quantity counted thus far, one has to interpret the recurrence in order to articulate. Thus, counting involves two processes identified in different neuronal architectures: recurrence and gating. Counting, like subitization, appears to involve a number of specialised areas in the brain. In counting we have a visual object recognition module, a cardinality representation module and a word-association module for verbal output. In addition, these modules interact with a short-term memory (module) to keep track of the counting process.

Recurrence involves a small amount of transient memory in the system. Gating is required for a system to learn which of the two networks is allowed to output. In SCOUSYST we use both recurrence and gating to show how objects arranged in a line may be counted through both an indication act and articulation. There is no direct comparison of our work in existing neural network literature; Amit's (1989) work is on number comprehension, but not number articulation, as with the majority of other works reported in this paper, whereas Ma and Hirai's (1989) work concentrates on number articulation, but not counting. Again, we have shown in SCOUSYST that by having a multi-net architecture one can simulate aspects of children's counting such as errors, and show how the errors decrease with age.

Finally, it is important to remember that whatever the origin of knowledge in the human brain, whether innate or learnt, subitization exists in adults and is used for numerical processing when time is short, even though the alternate counting mechanism is also present. It appears that both processes, subitization and counting, compete to process numbers, but that the human brain has learnt to gate the output of one of the networks depending upon the amount of time available to do the numerical processing. This concept is the focus of future work, whereby both SSUBSYST and SCOUSYST are to be combined into a single coherent multi-net system for quantification where the choice of processing path – to subitize or to count – is selected by time constraints imposed upon the simulation.

References

Abidi, S.S.R. & Ahmad, K. (1997). Conglomerate Neural Network Architectures: The Way Ahead for Simulating Early Language Development. *Journal of Information Systems Engineering*, vol. 13(2), pp. 235-266.

Ahmad, K., & Bale, T.A, (2001) Simulation of Quantification Abilities Using a Modular Neural Network Approach. *Neural Computing and Applications*, vol. 10(1), pp. 77-88.

Amari, S. (1980). Topographic Organization of Nerve Fields. *Bulletin of Mathematical Biology*, vol. 42, pp. 339-364.

Amit, D.J. (1989). *Modelling Brain Function: The World of Attractor Neural Networks*. Cambridge, UK: Cambridge University Press.

Amit, D.J. (1988). Neural Networks Counting Chimes. *Proceedings of the National Academy of Sciences*, USA, vol. 85, pp. 2141-2145.

Anderson, J.A. (1995). *An Introduction to Neural Networks*. Cambridge, MA.: MIT Press.

Anderson, J.R. (1993). *Rules of the Mind*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Anderson, J.A., Spoehr, K.T. & Bennett, D.J. (1994). A Study in Numerical Perversity: Teaching Arithmetic to a Neural Network. In Levine, D.S. & Aparicio, M. (Eds), *Neural Networks for Knowledge Representation and Inference*, pp 311-335. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Brannon, E.M. (2002). The Development of Ordinal Numerical Knowledge in Infancy. *Cognition*, vol. 83, pp. 223-240.

Brannon, E.M. & Terrace, H.S. (1998). Ordering of the Numerosities 1-9 by Monkeys. *Science*, vol. 282, pp. 746-749.

Butterworth, B. (1999). *The Mathematical Brain*. London: Macmillan.

Cipolotti, L. & Butterworth, B. (1995). Toward a Multiroute Model of Number Processing: Impaired Number Transcoding with Preserved of Calculation Skills. *Journal of Experimental Psychology: General*, vol. 124(4), pp. 375-390.

Cohen, L.B. & Chaput, H.H. (2002). Connectionist Models of Infant Perceptual and Cognitive Development. *Developmental Science*, vol. 5(2), pp. 173-174.

Dallaway, R. (1994). *Dynamics of Arithmetic: A Connectionist View of Arithmetic Skills*. Cognitive Science Research Papers 306. Brighton, UK: University of Sussex.

Dehaene, S. (2000). The Cognitive Neuroscience of Numeracy: Exploring the Cerebral Substrate, the Development, and the Pathologies of Number Sense. In Fitzpatrick, S.M. & Bruer, J.T. (Eds), *Carving Our Destiny: Scientific Research faces a New Millennium*, pp. 41-76. Washington: Joseph Henry Press.

Dehaene, S. (1997). *The Number Sense: How the Mind Creates Mathematics*. London: Allen Lane, The Penguin Press.

Dehaene, S. (1992). Varieties of Numerical Abilities. In Dehaene, S. (Ed), *Numerical Cognition* (1993), pp. 1-42. Cambridge, MA.: Blackwell Publishers.

Dehaene, S. & Changeux, J.P. (1993). Development of Elementary Numerical Abilities: A Neuronal Model. *Journal of Cognitive Neuroscience*, vol. 5(4), pp. 390-407.

Dehaene, S. & Mehler, J. (1992). Cross-Linguistic Regularities in the Frequency of Number Words. *Cognition*, vol. 43, pp. 1-29.

Elman, J. (1990). Finding Structure in Time. *Cognitive Science*, vol. 14, pp. 179-211.

Fahlman, S.E. & Lebiere, C. (1990). The Cascade-Correlation Learning Architecture. In Touretzky, D.S. (Ed). *Advances in Neural Information Processing Systems*, vol. 2, pp.524-532. San Mateo, CA.: Morgan Kaufmann.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position. *Pattern Recognition*, vol. 15(6), pp. 455-469.

Fuson, K.C. (1988). *Children's Counting and Concepts of Number*. Berlin, Heidelberg, New York: Springer-Verlag.

Fuson, K.C., Richards, J. & Briars, D.J. (1982). The Acquisition and Elaboration of the Number Word Sequence. In Brainerd, C.J. (Ed), *Children's Logical and Mathematical Cognition: Progress in Cognitive Development Research*, pp. 33-92. Berlin, Heidelberg, New York: Springer-Verlag.

Gallistel, C.R. & Gelman, R. (1992). Preverbal and Verbal Counting and Computation. In Dehaene, S. (Ed), *Numerical Cognition* (1993), pp. 43-74. Cambridge, MA.: Blackwell Publishers.

Gelman, R. & Gallistel, C.R. (1978). *The Child's Understanding of Number*. Cambridge, MA.: Harvard University Press.

Gelman, R. & Meck, E. (1983). Preschoolers' Counting: Principles Before Skill. *Cognition*, vol. 13, pp. 343-359.

Giles, C. L. & Maxwell, T. (1987). Learning, Invariance, and Generalization in High-order Neural Networks. *Applied Optics*, vol. 26(23), pp. 4972-4978.

Hoekstra, J. (1992). Counting with Artificial Neural Networks: An Experiment. In Aleksander, I. & Taylor, J. (Eds.), *Artificial Neural Networks*, vol. 2, pp. 1311-1314.

Humphreys, G.W. (1998). Neural Representation of Objects in Space: A Dual Coding Account. *Philosophical Transactions of the Royal Society of London, Series B-Biological Sciences*, vol. 353(1373), pp. 1341-1351.

Jacobs, R.A., Jordan, M.I. & Barto, A.G. (1991). Task Decomposition through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks. *Cognitive Science*, vol. 15, pp. 219-250.

Jevons, W.S. (1871). The Power of Numerical Discrimination. *Nature*, vol. 3, pp. 44-64.

Kaufman, E.L., Lord, M.W., Reese, T.W. & Volkmann, J. (1949). The Discrimination of Visual Number. *American Journal of Psychology*, vol. 62, pp. 498-525.

Kirby, K.N. (1992). Intensity of Stimulation, Necessary Truths, and the Acquisition of Numeracy. *Mind and Language*, vol. 7(4), pp. 359-363.

Kohonen, T. (1997). *Self-Organizing Maps*. 2nd Ed. Berlin, Heidelberg, New York: Springer-Verlag.

Ma, Q. & Hirai, Y. (1989). Modeling the Acquisition of Counting with an Associative Network. *Biological Cybernetics*, vol. 61, pp. 271-278.

MacWhinney, B. (1991). *The CHILDES Project: Tools for Analysing Talk*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Mandler, G. & Shebo, B.J. (1982). Subitizing: An Analysis of its Component Processes. *Journal of Experimental Psychology: General*, vol. 111, pp. 1 –22.

Mareschal, D. & Johnson, S.P. (2002). Learning to Perceive Object Unity: A Connectionist Account. *Developmental Science*, vol. 5(2), pp. 151-172.

Mareschal, D. & Shultz, T.R. (1999). Development of Children's Seriation: A Connectionist Approach. *Connection Science*, vol. 11(2), pp. 149-186.

McClelland, J.L. (1995). A Connectionist Perspective on Knowledge and Development. In Simon, T.J. & Halford, G.S. (Eds). *Developing Cognitive Competence: New Approaches to Process Modelling*, pp. 157-204. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

McCloskey, M., Caramazza, A. & Basili, A. (1985). Cognitive Mechanisms in Number Processing and Calculation: Evidence from Dyscalculia. *Brain and Cognition*, vol. 4, pp. 171-196.

McCloskey, M. & Lindemann, A.M. (1992). MATHNET: Preliminary Results from a Distributed Model of Arithmetic Fact Retrieval. In Campbell, J.I.D. (Ed). *The Nature and Origins of Mathematical Skills*, pp. 365-409. North Holland: Elsevier Science Publishers B-V.

Meck, W.H. & Church, R.M. (1983). A Mode Control Model of Counting and Timing Processes. *Journal of Experimental Psychology: Animal Behavior Processes*, vol. 9(3), pp. 320-334.

Nye, J., Clibbens, J. & Bird, G. (1995). Numerical Ability, General Ability and Language in Children with Down’s Syndrome. *Down's Syndrome: Research and Practice*, vol. 3, pp. 92-102.

Peterson, S.A. & Simon, T.J. (2000). Computational Evidence for the Subitizing Phenomenon as an Emergent Property of the Human Cognitive Architecture. *Cognitive Science*, vol. 24(1), pp. 93-122.

Piaget, J. (1952). *The Child’s Conception of Number*. London: Routledge & Kegan Paul Limited.

Rodriguez, P., Wiles, J., Elman, J.L. (1999). A Recurrent Neural Network that Learns to Count. *Connection Science*, vol. 11(1), pp. 5-40.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning Internal Representations by Error Propagation. In Rumelhart, D.E. & McClelland J.L. (Eds) (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1: Foundations, pp. 318-362. Cambridge, MA.: MIT Press.

Sharkey, A.J.C. (Ed) (1999). *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*. Berlin, Heidelberg, New York: Springer-Verlag.

Sophian, C. & Adams, N. (1987). Infants' Understanding of Numerical Transformations. *British Journal of Developmental Psychology*, vol. 5, pp. 257-264.

Spelke, E. (1994). Initial Knowledge: Six Suggestions. *Cognition*, vol. 50, pp. 431-445.

Strauss, M.S. & Curtis, L.E. (1984). Development of Numerical Concepts in Infancy. In Sophian, C. (Ed). *Origins of Cognitive Skills*, pp. 131-155. Hillsdale, NJ: Erlbaum.

Thompson, R.F., Mayers, K.S., Robertson, R.T. and Patterson, C.J. (1970). Number Coding in Association Cortex of the Cat. *Science*, vol. 168, pp. 271-273.

Trick, L., & Pylyshyn, Z. (1994). Why are Small and Large Numbers Enumerated Differently? A Limited-capacity Preattentive Stage in Vision. *Psychological Review*, vol. 101, pp. 80-102.

Willshaw, D.J. & von der Malsburg, C. (1976). How Patterned Neural Connections can be set up by Self-Organization. *Proceedings of the Royal Society, Series B*, vol. 194, pp. 431-445.

Wynn, K. (1995). Origins of Numerical Knowledge. *Mathematical Cognition*, vol. 1(1), pp. 35-60.

Wynn, K. (1992a). Evidence Against Empiricist Accounts of the Origins of Numerical Knowledge. *Mind and Language*, vol. 7(4), pp. 315-332.

Wynn, K. (1992b). Issues Concerning a Nativist Theory of Numerical Knowledge. *Mind and Language*, vol. 7(4), pp. 367-381.

Acknowledgements

Tracey Bale gratefully acknowledges the support of UK Engineering and Physical Sciences Council in terms of a PhD Studentship (1994-97) and that of the EU sponsored ACE Project (ESPRIT Project No. 22271, 1995-1998) from February to December 1998. The authors would also like to thank the three anonymous reviewers of this article for their comments.