Chapter 3

In: Y Munakata and MH Johnson, eds. *Processes of change in brain and cognitive development: Attention and performance XXI*, pp. 61-86. Oxford: Oxford University Press.

Constructive learning in the modeling of psychological development

Thomas R. Shultz

Abstract

Although many computational models of psychological development involve only learning, this paper examines the advantages of allowing artificial neural networks to grow as well as learn in such simulations. Comparisons of static and constructive network simulations of the same developmental phenomena bring this issue into clear focus. Constructive algorithms are found to be better at learning and at covering developmental stages and stage sequences. The relatively new sibling-descendant cascade-correlation algorithm (SDCC) decides whether to install each newly recruited hidden unit on the current highest layer or on a new layer. SDCC is applied to the problem of conservation acquisition in children. Results show no differences in comparison to previous conservation simulations done with standard cascade-correlation except for fewer network layers and connections with SDCC.

3.1 Introduction

It is clear from publication counts that artificial neural networks have become the model of choice in developmental psychology, just as they have in several other psychological domains (Shultz 2003). The most popular type of neural network in psychological modeling, including developmental work, is the static feed-forward network. Such networks learn by adjusting their connection weights, but their initial topologies do not change across the developmental periods that they model. It is development by learning, with a dose of nativism thrown in because the topological structure of each network is fully specified at the start.

A principal alternative is constructive learning, which, like psychological constructivism, allows for qualitative growth as well as quantitative adjustments. In some theoretical approaches, development proceeds in a succession of stages. Constructive learning fits well with those stage theories emphasizing that cognitive systems themselves undergo significant changes during development (e.g. Piaget 1965).

Constructive learning is also consistent with contemporary neuroscience findings of synaptogenesis and neurogenesis occurring under pressures to learn, not only in infancy but also in mature brains (cf. reviews by Gould *et al*. 1999; Kempermann and Gage 1999; Purves *et al*. 1996; Quartz 2003; Quartz and Sejnowski 1997; Shultz *et al*. 2006). This line of neuroscience research is developing rapidly and some of it, including the claim of learning-directed neurogenesis in primate cortex, is controversial. Synaptogenesis is much less controversial, and it is worth noting that the constructive algorithms discussed here are neutral with respect to whether their growth processes emulate synapto- or neurogenesis. The issue hinges on whether candidate recruits are already somewhere else in the system or are constructed afresh.

There are several computational arguments showing the advantages of constructive learning over static learning. First, constructive-network algorithms learn in polynomial time any problem that can be learned in polynomial time (i.e. a realistic amount of time) by any algorithm whatsoever (Baum 1989). In contrast, static-network algorithms may take considerably longer to master such problems (e.g. exponential time), or maybe never solve them, because there is no guarantee that a given network topology has been designed with the appropriate degree of computational power. A designer of a static network must find a network topology with the approximately correct degree of power. Static networks that are too weak fail to learn and those that are too strong fail to generalize. Constructive algorithms automatically try to find the right degree of network complexity for the problem being learned. At the same time they are searching in weight space, they are also searching in topology space.

Second, constructive learners may find optimal solutions to the bias/variance tradeoff that plagues static learners (Quartz 2003; Quartz and Sejnowski 1997). All learning algorithms face a potential tradeoff between bias and variance, both sources of error. Bias is the distance between the best hypothesis that a learning system can form and the correct hypothesis; variance is the distance of a learning system's current hypothesis from the correct hypothesis. An optimal learning system would manage to keep both bias and variance low, and thus finish close to the correct hypothesis. A small network restricts variance but is highly biased because it can express only a few hypotheses, all of which may be far from the correct hypothesis, thus contributing to error. In contrast, a large network reduces bias by expanding the hypothesis space, but also typically increases variance; because so many hypotheses are possible, the correct hypothesis may be difficult to find, keeping error high. Constructive algorithms may avoid this tradeoff by reducing bias (by adding hidden units to expand the network and the hypothesis space) while reducing variance (by adjusting connection weights to approach the correct hypothesis).

Third, constructive, but not static, learners can escape from Fodor's paradox that nothing genuinely novel can be learned (Mareschal and Shultz 1996; Quartz 1993). Fodor (1980) had argued, based on 1970s hypothesize-and-test learning algorithms, that a learning system can only learn what it can already represent. If a particular hypothesis cannot be represented then it cannot be tested, and if it cannot be tested then it cannot be learned. Constructive learners escape from this dilemma because their growth enables them to represent hypotheses that they could not represent previously with their limited computational power. In contrast, static learning systems are limited to those hypotheses permitted by their initial design. A static network with a particular number of units and a

particular connection scheme may not be able to adjust its weights appropriately, no matter how favorable its training regime is; the network may be too weak to learn, too powerful to generalize, or may not have the right topology for the problem at hand.

In this paper, I explore the advantages that constructive learning algorithms have in modeling various phenomena in psychological development, a strategy that is distinct from, but supported by, the foregoing neurological and computational considerations.

3.2 Comparison of constructive and static networks

One way to highlight the unique benefits of constructive learning is to compare models whose networks are allowed to grow against models containing static networks. The neural algorithms most widely applied to simulating cognitive development are back-propagation (BP), a static learner, and cascade-correlation (CC), a constructive learner (Shultz 2003). Both of these algorithms employ feed-forward, multilayered networks that learn by adjustment of connection weights to reduce error. Unlike BP, CC grows its network structure by recruiting hidden units (Fahlman and Lebiere 1990). Neither BP nor CC is designed to model neural circuits in detail; they are instead functional algorithms inspired by neuroscience principles. Mathematical and computational details on these algorithms can be found in many other sources (e.g. Shultz 2003).

Both BP (Elman *et al*. 1996; Munakata and McClelland 2003; Shultz 2003) and CC (Shultz 2003) have been successfully applied to a wide range of developmental phenomena. Nonetheless, comparisons between BP and CC are difficult to make because these algorithms have most often been applied to distinct phenomena. A unique focus of the present paper is a series of direct *bakeoff* competitions in which CC and BP are applied to the same phenomena. Although common and valuable in machine learning research, simulation bakeoffs are still rare in developmental psychology.

There are three domains of cognitive development to which both of these algorithms have been applied: the balance scale, the integration of velocity, time, and distance cues for moving objects, and age changes in sensitivity to correlations vs. features in category learning by infants. These three competitions are reviewed before presenting a new bakeoff featuring static and constructive models of conservation acquisition.

3.2.1 Balance scale

The balance-scale problem presents a child with rigid beam balanced on a fulcrum (Siegler 1976). The beam has several pegs spaced at regular intervals to the left and right of the fulcrum. The experimenter places some number of identical weights on a peg on the left side and some number of weights on a peg on the right side. While supporting blocks prevent the beam from moving, the child predicts which side of the beam will descend, or whether the scale will balance, when the supporting blocks are removed.

The rules children use to make such predictions can be diagnosed by presenting children with six different types of problems. Three of these problems are simple because one cue (weight or distance from the fulcrum) perfectly predicts the outcome while the other cue is constant on both sides of the scale. The other three problems are more complex because these two cues conflict with each other, weight predicting one outcome and distance predicting another outcome. The pattern of predictions across these six problem types can be used to diagnose the rule the child uses to guide predictions. In a

connectionist network, such rules are not symbolic propositions, but are instead emerging epiphenomena of network structure and function.

The major psychological regularity in the balance-scale literature is progression through four different stages (Siegler 1976). In stage 1, children use weight information to predict that the side with greater weight will descend or that the scale will balance when the two sides have equal weights. In stage 2, children also begin to use distance information when the weights are equal on each side, predicting that in such cases the side with greater distance will descend. In stage 3, weight and distance information are emphasized equally, but the child guesses when weight and distance information conflict on the complex problems. In stage 4, children respond correctly on all types of problems, whether simple or complex.

In perhaps the first connectionist simulation of cognitive development, McClelland and Jenkins (1991) found that a static BP network with two groups of hidden units segregated for either weight or distance information developed through the first three of these stages and into the fourth stage. However, their networks did not settle in stage 4, instead continuing to cycle between stages 3 and 4. Subsequent simulations showed that BP networks could settle into stage 4, but only at the cost of skipping stages 1 and 2 (Shultz 2003). In contrast, the first CC model of cognitive development naturally captured all four balance-scale stages, without requiring segregation of hidden units (Shultz *et al*. 1994).

The ability of neural networks to capture stages 1 and 2 on the balance scale is due to a bias towards equal-distance problems in the training set. These are problems with the weights placed equally distant from the fulcrum on each side. A preponderance of equal-distance problems forces the network to emphasize weight information first because weight information is more relevant to reducing network error. Once weight-induced error is reduced, then the network can turn its attention to distance information. In effect, the learning algorithm must find the particular region of connection-weight space that allows it to emphasize the numbers of weights on the scale before moving to another region of weight space that allows multiplication of equally important weight and distance information. A powerful learning algorithm is required to make this move in connection-weight space. Apparently a static network, once committed to using weight information in stage 1, cannot easily find its way to a stage-4 region merely by continuing to reduce error. In contrast, a constructive algorithm has an easier time with this move because each newly recruited hidden unit changes the shape of connection-weight space by adding a new dimension to this space. A new dimension provides a path on which to move towards a stage-4 region. The fact that several other algorithms fail to reach stage 4 on the balance scale means that the CC model is somewhat unique in predicting the occurrence of stage 4, a stage consistently found in psychological research.

Our CC networks also predicted the other major psychological regularity in the balance-scale literature, the so-called *torque-difference* effect, which is that problems with large torque differences from one side of the scale to the other are easier for children to solve at every stage than problems with small torque differences. We subsequently discovered that this somewhat ignored phenomenon had already been noted with children and could also be obtained from the BP model (Shultz 2003).

3.2.2 Integration of velocity, time, and distance cues

In classical physics, the velocity of a moving object equals the ratio of distance traveled to the time of the journey: velocity = distance / time. Algebraic manipulations show that distance = velocity x time, and time = distance / velocity. Some of the clearest evidence on children's acquisition of these ideas was provided by Wilkening (1981), who asked children to predict one dimension from knowledge of the other two. For example three levels of velocity information were represented by motion of a turtle, a guinea pig, and a cat. These three animals were said to be fleeing from a barking dog, and children were asked to imagine these animals moving while the dog barked. The child's task was to infer how far an animal would travel given the duration of the barking, an example of inferring distance from velocity and time.

We diagnosed rules in CC simulations based on correlations between network outputs and the various algebraic rules observed in children (Buckingham and Shultz 2000). To be diagnosed, an algebraic rule had to correlate positively with network responses, account for most of the variance in network responses, and account for more variance than any other rules did. For velocity and time inferences, CC networks first developed an equivalence rule (e.g., velocity = distance), followed by a difference rule (e.g. velocity = distance - time), followed in turn by the correct ratio rule (e.g. velocity = distance / time). Results were similar for distance inferences, except for the absence of an equivalence rule. For distance inferences, there is no reason for a network to favor either velocity or time information because both vary positively with distance. Six of these stages had been previously found with children, and two others were predictions of CC network models subsequently confirmed with children. The two predicted stages were the final stage of velocity inferences (velocity = distance / time) and the first stage of time inferences (time = distance). Stage progressions resulted from continued recruitment of hidden units.

In contrast to constructive networks, static networks were unable to capture these stage sequences (Buckingham and Shultz 1996). If a static BP network has too few hidden units, it fails to reach the correct multiplicative rules; if it has too many hidden units, it fails to capture the intermediate additive (difference) stages on velocity and time inferences. Extensive exploration of a variety of network topologies and variation in learning parameters suggests that there is no static network topology capable of capturing all three stages in this domain. Even the use of cross-connections that bypass hidden layers failed to improve the stage performance of BP networks. There is no apparent way to get static networks to cover all stages because the difference between underpowered and overpowered BP networks is a single hidden unit. Thus, the ability to grow in computational power seems essential in simulating stages in this domain.

3.2.3 Age changes in early category learning

A third comparison concerns age changes in sensitivity to correlations vs. features in category learning by infants. Using a familiarization paradigm to study categorization, Younger and Cohen (1983, 1986) found that 4-month-olds process information about independent features of visual stimuli, whereas 10-month-olds additionally abstract relations among those features. These results relate to a classic controversy about the extent to which perceptual development involves integration or differentiation of stimulus information. Developing ability to understand relations among features suggests that

perceptual development involves integration, a view compatible with constructive learning.

Infants are conventionally assumed to construct representational categories for repeated stimuli, and afterwards to ignore novel stimuli that are consistent with a category, while concentrating on novel stimuli that are not members of an existing category. After repetitions of visual stimuli with correlated features, 4-month-olds recovered attention to stimuli with novel features more than to stimuli with either correlated or uncorrelated familiar features (Younger and Cohen 1983, 1986). In contrast, 10-month-olds recovered attention both to stimuli with novel features and to stimuli with familiar uncorrelated features more than to stimuli with familiar correlated features. This pattern of recovery indicates that young infants learned about individual stimulus features, but not about the relationships among features, whereas older infants also learned how these features correlate with one other.

These phenomena were recently simulated with CC encoder networks (Shultz and Cohen 2004). Encoder networks have the same number of inputs as outputs. Their task is to reproduce their input values on their output units. Encoder networks learn to do this by encoding an input representation onto a relatively small number of hidden units and then decoding that representation onto output units. Such networks develop a recognition memory for the stimuli they are exposed to. Network error, the discrepancy between inputs and outputs, can be used as a measure of stimulus novelty. Both static and constructive encoder networks were applied to these phenomena (Shultz and Cohen 2004). In CC networks, age was implemented by setting the score-threshold parameter higher for 4-month-olds than for 10-month-olds. Because learning stops only when all network outputs are within score-threshold of their target values for all training patterns, this parameter governs how much learning occurs. We assumed that older infants learn more from the same exposure time than younger infants do. On the further assumption that network error reflects stimulus novelty and interest, CC networks covered the infant data by showing more error to the uncorrelated test stimulus than to the correlated test stimulus only at the smaller score-threshold.

In contrast, a wide range of static BP networks failed to capture these effects. A BP simulator was modified to use a score-threshold parameter to decide on learning success like CC does. A wide variety of score-threshold values were explored in a systematic attempt to get BP networks to cover the infant data. Initially BP networks were given three hidden units, the modal number recruited by CC networks on this task. BP network topology was then varied to explore the roles of network depth and the presence of cross connections in simulation success. Both flat and deep BP networks were tried, both with and without cross-connection weights. Although BP networks did show some learning, they covered neither the 4-month-old nor 10-month-old data nor the transition between these ages. We ran nine BP networks in each of 80 simulations (2 familiarization sets x 10 score-threshold levels x 4 network topologies).

Only the CC networks generated a crossover prediction, with deep learning showing a correlation effect, and very superficial learning showing the opposite – a similarity effect, in the form of more error to (or interest in) the correlated test item than to the uncorrelated test item. This is termed a *similarity* effect because the uncorrelated test item was most similar to those in the familiarization set. The simulation predicted a correlation effect under optimal learning conditions and a similarity effect with less than optimal

familiarization learning. Tests of this prediction found that 10-month-olds who tired of the training stimuli looked longer at the uncorrelated than the correlated test stimulus, but those who did not so tire did the opposite, looking longer at the correlated than the uncorrelated test stimulus (Cohen and Arthur 2004).

Although the static network simulations did not cover the infant data, they were useful in establishing why constructive networks were successful because these static networks employed topologies copied from the final constructive networks. Network growth seems critical to capturing detection of correlations between features and the developmental shift from earlier feature-value learning. An initially underpowered network focuses only on the stimulus features, allowing later network growth to use those features in detecting correlations.

3.3 Conservation phenomena

Conservation is one of the most well-studied phenomena in cognitive development with over 1000 publications. It most often involves a belief in the continued equivalence of two physical quantities over a transformation that appears to alter one of them. A prime example of conservation presents a child with two identical rows of evenly spaced objects. After the child agrees that the two rows have the same number of objects, the experimenter transforms one of the rows, for example by spreading its items out. The child is then asked whether the two rows still have the same amount or whether one of them now has more. Children below about six years of age respond that one of the two rows, usually the longer row, now has more than the other (Piaget 1965). This is in contrast to older children who respond that the two rows still have equal amounts. That is, older children conserve equivalence of the two amounts over the compressing transformation.

In the vast literature on conservation, there are four well-known empirical regularities:

1. A shift from non-conservation to conservation beliefs (acquisition effect).
2. A sudden spurt in performance during acquisition (spurt effect).
3. Emergence of correct conservation judgments for small quantities before larger quantities (problem-size effect).
4. Non-conservers' choice of the longer row as having more items than the shorter row (length-bias effect).

Regularities 1, 3, and 4 have been replicated many times with children (cf. Shultz 1998 for a more complete review).

3.3.1 Simulations of conservation with CC networks

Simulations with CC networks covered all of these phenomena without having to bias the training environment in any particular way (Shultz 1998). Inputs to the networks described conservation problems in which rows of objects were coded for their perceptual characteristics, namely their length and density. Transformations included those that alter number (addition and subtraction) and those that preserve number (elongation and compression). Addition and subtraction transformations each changed the quantity of a row by one item. Elongation and compression transformations altered the density of the row by one level. Target feedback supplied to the network concerned relative numerical equality judgments comparing the two rows. There were 5 levels of length and 5 levels of density, ranging from 2 to 6 in the initial rows, which could be either equal or unequal in

number. Reflecting physical and mathematical reality, the number of items equals length x density of the row. Table 3.1 shows some example transformations of a 6-item row with initial length of 3 and initial density of 2. Typical psychology experiments, starting with only equal rows and using only equality-preserving transformations, can be seen as a special case of this more general scheme. This more general scheme, however, is necessary to capture what children eventually come to know about conservation problems.

Table 3.1 Example transformations holding density constant when number is altered

Transformation	Length	Density	Number
Pre-transformation	3	2	6
Add	3.5	2	7
Subtract	2.5	2	5
Elongate	6	1	6
Compress	2	3	6

Length and density values were coded as real numbers for each of the rows, both before and after the transformation. The identity of the transformed row was coded on a binary unit as 1 or -1. The four transformation types were coded in a binary, nth-unit fashion as 1 for the unit corresponding to the transformation type and -1 on the other three units. Two target output units with sigmoid activation functions were coded as (0.5 -0.5) if the first row had more items, (-0.5 0.5) if the second row had more items, and (-0.5 -0.5) if the two rows had equal numbers of items after the transformation. Thus the networks had as inputs the same information that children in conservation experiments have regarding how the rows look, both before and after the transformation, as well as which transformation was applied and the row it was applied to. What the networks had to learn are functions relating these inputs to a decision about which row has more or whether the rows have the same number.

For each network, 420 of the 600 possible conservation problems were randomly selected as training patterns and 100 others were randomly selected as test patterns. Because networks started from randomly-selected connection weights and learned in a randomly-designed environment, they differed from each other in both heredity and environment, something like children do. Training was stopped at 1500 epochs, by which time most networks had learned all of the training patterns, a state called ‘reaching victory’. A single epoch is a pass through all of the training patterns. Testing of a network was done on its 100 test patterns not used in training. All four of the principal conservation phenomena were captured by these CC networks.

3.3.2 Simulations of conservation acquisition with SDCC networks

The CC algorithm has occasionally been criticized for constructing overly deep networks that may not generalize as well as flatter networks do. The reason CC networks are so deep is that each new hidden unit is installed on its own layer with cascaded weights from any previous hidden units. A variant called flat CC puts all hidden units on the same layer and eliminates the cascaded weights between these hidden units, in the fashion of a three-layered BP network. In a research design that had flat and standard CC networks learning

either flat or deep problems, we found that, on simpler problems (with less than 20 hidden units), there were no differences between flat and standard CC. On more complex problems (with more than 20 hidden units), there were several differences: 1) each algorithm performed better on its own problem type than on the opposite problem type, 2) standard CC learned more efficiently and accurately than flat CC, and 3) flat CC required fewer weights and generalized better than standard CC.

A relatively new variant called sibling-descendant CC (SDCC) dynamically decides whether to install a new hidden unit on the current highest layer (as a sibling) or on a separate, higher layer (as a descendant) (Baluja and Fahlman 1994). In SDCC, the candidate pool (conventionally eight candidate units) is divided in half, with an equal number of sibling and descendant candidates. The single candidate whose activations correlate most highly with network error is the one actually recruited, following conventional CC policy. To counter the natural tendency to always recruit a descendant unit, the descendant correlation values are multiplied by a penalty factor. Empirical exploration discovered that a penalty factor of 0.8 limited network depth without harming generalization (Baluja and Fahlman 1994).

Systematic comparisons show that SDCC's performance lies somewhere between flat and standard CC. There are generally no significant differences between SDCC and standard CC on most performance measures. The only exception is that SDCC generates networks with fewer layers than does standard CC. Hence our interest in applying SDCC in simulations of psychological data with the aim of building networks with more realistic depths than in standard CC.

3.3.2.1 Acquisition of conservation

Twenty SDCC networks were trained and tested in the same fashion as CC networks were. In all SDCC simulations reported here, the penalty factor for descendant units was 0.8. A wide range of hidden-unit topologies were spontaneously constructed as shown in Table 3.2. For example, a topology of 2-4-4 means two hidden units on the first layer, four on the second layer, and four on the third layer. All of these networks had 13 input units and two output units.

Table 3.2 Hidden-unit topologies of 20 SDCC networks

2-4-4	1-3-5	2-2-2-1	2-3-1-3	1-3-2-2-1-1
2-3-6	2-7	3-3-3	1-1-5-1	2-3-2
5-3	1-1-5-3	2-2-3-1	2-3	2-2
3-1-1-1	4-3-3	3-1-2-2	7-1	3-2-1

To compare overall performance of these SDCC networks with ordinary CC, a fresh experiment was run with 20 CC networks, a replication of the first experiment in Shultz (1998). Table 3.3 presents mean results on several variables of interest, along with independent-sample *t*-tests. The two algorithms do not differ on any measure except for the obvious one of the number of layers of hidden units. Because SDCC often installs sibling hidden units, it used fewer layers and saved a mean of 4.25 connection weights per network, the saved weights being cascaded weights eliminated for hidden units installed on the same layer. These results establish that SDCC performs just as effectively as CC on conservation acquisition, while saving on both network depth and number of

connection weights. In the following sections, whether SDCC covers other conservation phenomena as well as CC does will be evaluated.

Table 3.3 Mean performance of 20 CC and 20 SDCC networks on conservation acquisition

Variable	CC	SDCC	*t*(38)	*p* <
Train % correct	99.13	99.24	-0.24	ns
Test % correct	93.35	94.20	-0.47	ns
Train error	0.008	0.009	-0.38	ns
Test error	0.041	0.039	0.23	ns
Epochs to victory	1257	1292	-0.40	ns
% reaching victory	50	65	-0.95	ns
Hidden units	7.95	8.10	-0.23	ns
Layers of hidden units	7.95	3.25	8.63	0.0001

Proportion of correct training and test patterns is shown in Fig. 3.1 for network 14. This network reached victory in 802 epochs, recruiting two hidden units in a first layer and three more in a second layer, for a final 13-2-3-2 topology. The output epochs are grouped into 100 equal-size blocks for the plot. The plot shows excellent generalization to the test patterns and a substantial spurt in performance, both of which are representative of all other CC and SDCC networks. The epochs at which hidden units were recruited are shown by triangles for descendant units and squares for sibling units. One unusual feature of this network is the close proximity of the second and third recruits. The second recruit, a sibling unit, did not improve performance so recruitment of another unit, a descendant unit, quickly followed. This and other hidden units continued to improve performance of the network to near perfect.

Fig 3.1 Acquisition of conservation by SDCC network 14.

3.3.2.2 Problem size

The problem-size effect is that children develop conservation with small numbers before large numbers. For the simulations, a problem was considered to be small if the number of the smaller row had less than 12 items; large if the number of the smaller row had more than 24 items. These particular values split the patterns so that small and large problems were about equally frequent. Proportions of small and large problems correct for the 20 networks are plotted in Fig. 3.2 over ten equal-size blocks of output epochs. Proportions were transformed to arcsins to uncorrelate the relation between means and variances, and these arcsin values were subjected to ANOVA in which epoch block and problem size served as repeated measures. There were main effects of size, $F(1, 19) = 15$, $p < 0.001$, and block, $F(9, 171) = 172$, $p < 0.0001$, and an interaction between them, $F(9, 171) = 8.7$, $p < 0.0001$. Problem-size means at each block were compared with dependent-sample t-tests. At every block except 7 and 10, the difference was significant, $p < 0.05$, although the direction was reversed at block 1 where understanding of conservation was still very poor. The magnitude of the problem-size effects in networks (0.1) is about the same as reported in children, at least over blocks 2-6 (cf. Shultz 1998).

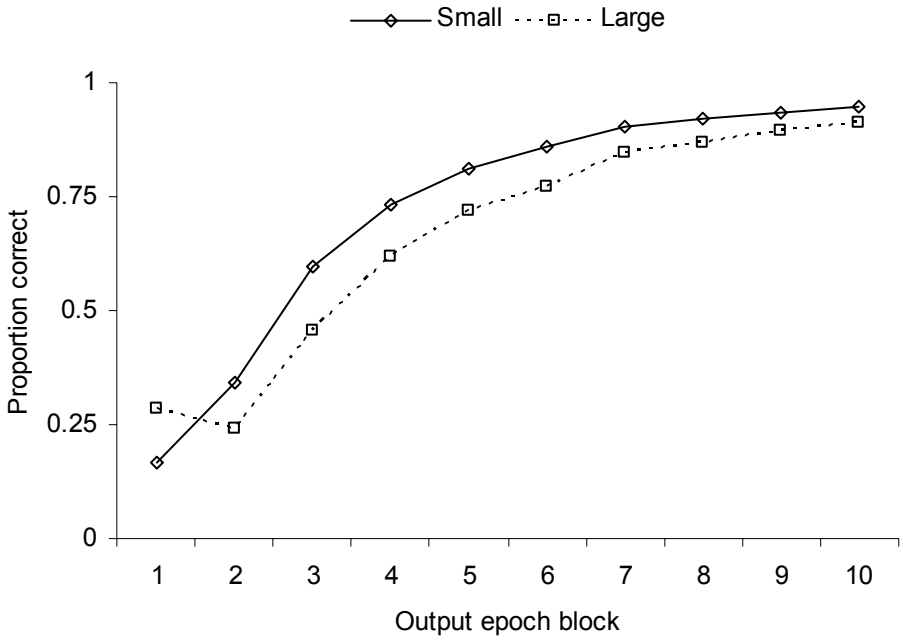

Fig 3.2 Mean proportion correct on small- and large-number conservation test patterns over ten equal-size blocks of output epochs for 20 SDCC networks.

The fact that the networks performed better on small-number problems than on large-number problems is a manifestation of the Weber-Fechner law, wherein larger proportional differences are easier to discriminate than smaller proportional differences. Such effects naturally fall out of neural networks, although one could have captured this effect in other, more obvious ways suggested in the psychological literature, namely, that children get relatively more practice with small numbers, or that children's estimation techniques are relatively more accurate with small numbers.

3.3.2.3 Length bias

Length bias, in the context of conservation of discrete quantities, refers to choosing a longer row as having more items than a shorter row. Analogous errors have been reported among non-conservers in a variety of other conservation tasks. The test for length bias conventionally uses only elongation and compression problems that have initially equal rows. Thirty of the 100 such equality-conserving problems were randomly selected as test problems. The remaining 70 equality-preserving problems entered the training set, conforming to a policy of training on 70 per cent of the relevant problems, along with 350 other randomly-selected training problems. The alternative of choosing the shorter, denser row as having more items in these equality-conserving problems is considered a density bias.

If the first output activation was more than 0 and the second output activation was less than 0, then a network was deemed to have chosen the first row as having more. Conversely, if the first output activation was less than 0 and the second output activation was greater than 0, then a network was considered to have chosen the second row as having more. Finally, if both output activations were less than 0, then a network was considered to have decided that the rows are numerically equal.

The numbers of test problems on which a network showed either a length or density bias each output epoch were collapsed into mean problems showing each bias at each of ten equal-size blocks of output epochs. These mean numbers of biased answers were subjected to ANOVA in which epoch block and type of bias served as repeated measures. There were main effects of bias, $F(1, 19) = 9.6$, $p < 0.01$, and block, $F(9, 171) = 12$, $p < 0.001$, and an interaction between them, $F(9, 171) = 4.9$, $p < 0.001$. Relevant means are shown in Fig. 3.3.

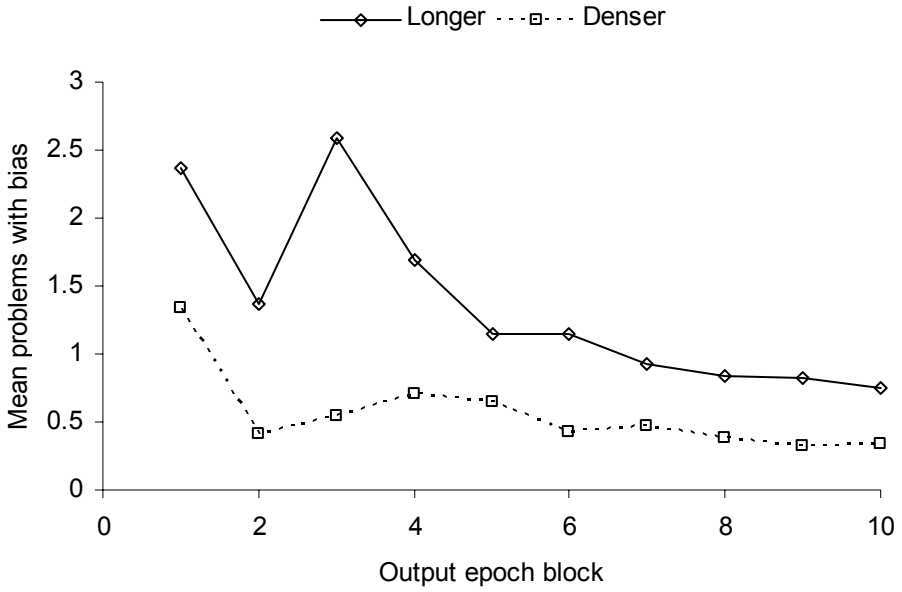

Fig 3.3 Length and density bias over blocks of output epochs in 20 SDCC networks.

Bias-type means at each block were compared with dependent-sample *t*-tests. Only at blocks 1-4 and 6 was this difference significant, $p < 0.05$. Although this length bias was

evident in the first two blocks, it increased in size in the third and fourth blocks before starting to diminish. This pattern suggests a bias that increases and then decreases with learning.

When expressed as a proportion of errors reflecting a length bias, such proportions ranged from 0.64 to 0.83 across the ten blocks. This approximates the values of 0.69 to 0.86 reported in various experimental conditions by Miller *et al*. (1973) with clay sausages, representing the proportion of length-bias errors in conservation of continuous quantities.

3.3.2.4 Explaining length bias

Length bias on conservation tasks has been conventionally attributed to either perceptual salience (length is more salient than number) or learning that longer rows often have more items than shorter rows. Although the source of length bias has not been settled by psychological research, it has been noted that very young children do not show a length bias (cf. review by Shultz 1998). This finding is consistent with the idea that time is required for children to learn that length is a somewhat reliable cue to number.

The learning explanation is attractive in that addition and subtraction transformations could introduce a correlation between number and length, as long as density is held constant during such transformations. When density is constant, adding an item makes a row longer as well as more numerous; subtracting an item makes a row shorter as well as less numerous. The impact of this correlation between length and number on conservation acquisition was tested by creating an alternate environment in which length, rather than density, is held constant during number-altering transformations. In this alternative environment, when an item is added to a row, the row is compressed so that density increases and length does not change. Likewise, during subtraction, a row is elongated so that density decreases and length again remains constant. To maintain consistency with the previous simulation, length rather than density is changed by one unit in elongation and compression transformations. With this change, the previous simulation was repeated with 20 fresh networks.

Mean test problems showing length and density bias are plotted in Fig. 3.4 for this strange training environment. ANOVA yielded main effects of bias, $F(1, 19) = 10.5$, $p < 0.01$, and block, $F(9, 171) = 15$, $p < 0.001$, and an interaction between bias and block, $F(9, 171) = 5.5$, $p < 0.001$. Bias-type means at each block were again compared with dependent-sample *t*-tests, revealing significant density bias only in blocks 1-4, $p < 0.01$. As with length bias, density bias increases in strength and then disappears as the network continues to learn about conservation phenomena.

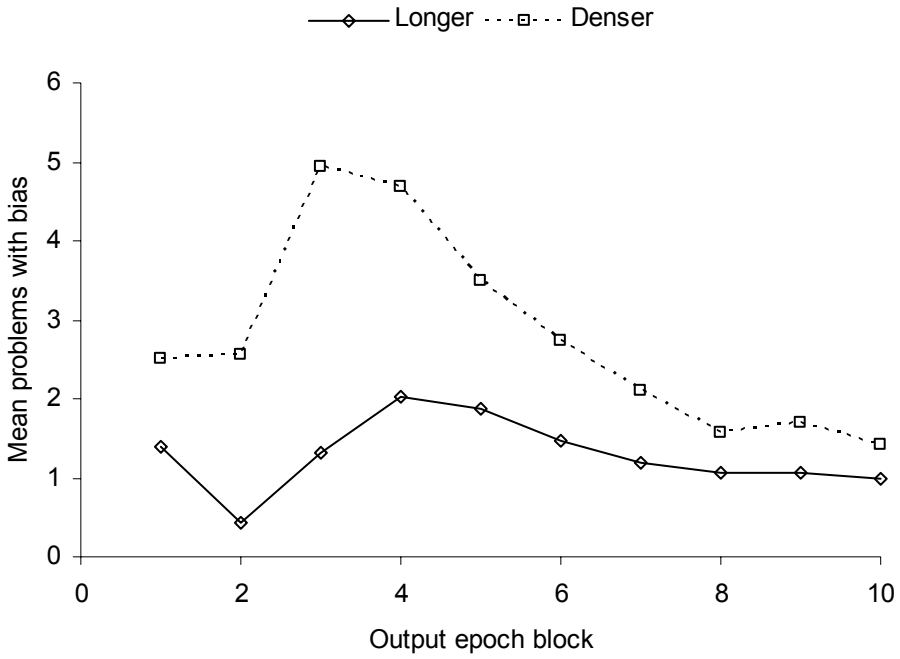

Fig 3.4 Length and density bias over blocks of output epochs in 20 SDCC networks trained in an environment in which length was held constant during number-altering transformations.

These results show that early non-conservation biases are strongly influenced in SDCC simulations by the correlation of either length or density with number during number-altering transformations. When density is held constant in these transformations, number correlates better with length than with density, and length then predicts number. But when length is held constant, number correlates with density, and thus density predicts number. Due to human laziness, the default condition is undoubtedly constant density because it takes more effort to adjust density when adding or subtracting an item.

These perceptual bias effects underscore the tension between perceptual and cognitive processes in conservation tasks. Piaget designed these tasks to create a particular conflict between perception and cognition. What the child knows (that a transformation does not alter a quantity) appears to conflict with what she sees (that one row is longer, and thus seems more numerous than the other row). It is well known that perceptual solutions dominate early in children's conservation acquisition, but eventually lose out to cognitive solutions involving reasoning about the effects of transformations (cf. review by Shultz 1998). Results in the last two simulations are consistent with this literature.

3.3.2.5 Knowledge-representation analysis

Although SDCC networks covered all four principal conservation phenomena (acquisition, spurt, problem size, and length bias), it would be premature to conclude that they provide an adequate model without analyzing the knowledge representations that they construct at various points in development. One of the most effective techniques for this is a Principal Components Analysis (PCA) of network contributions. Contributions are products of sending-unit activations and connection weights entering output units. As

such, contributions summarize all of the relevant information influencing a network's outputs and typically produce sensible abstract descriptions that generalize well across cross-connected networks that otherwise appear to be idiosyncratic in topology and connection weights (Shultz *et al*. 1995).

Table 3.4 shows the developmental history of knowledge representations in network 8. This network ended up with a 13-4-3-2 topology, that is with 13 input units, four hidden units on one layer, three hidden units on the next layer, and two output units. Without any hidden units, the PCA of contributions yields two components explaining 92 per cent of the variance in contributions. The first component has loadings from the four density inputs, and the second component has loadings from the four length inputs. At this point, the network is performing well below chance, being correct on only 19 per cent of the training problems and 8 per cent of the test problems. Because this network, like most networks, began near the chance level of 33 per cent correct, it is now well into the phase of non-conservation errors, consistently selecting a longer row as being more numerous than a shorter row.

Table 3.4 Developmental history of knowledge representations in conservation network 8

Hidden	Component						% Correct		% Variance
units	1	2	3	4	5	6	Train	Test	explained
0	dnst	lngth	-	-	-	-	19	8	92
1	dnst	lngth	h1, id, add, elng	-	-	-	40	44	85
2	lngth, h2	h1, sub	id, sub, elng	-	-	-	66	58	82
3	lngth, h2	h3, cmpr, dnst	h1, sub, id	h2, elng, h3	-	-	73	76	86
4	lngth, h2	h3, cmpr, dnst, add	h1, sub, id	h2, elng, h3	-	-	74	73	91
5	lngth, h2	cmpr, h3	h4, h5, add	h1, id	elng, h3, sub	-	94	90	89
6	lngth, h2	cmpr, h6, h3	h4, h5	id, h1	add, sub, h6, h1	elng, h3, sub	99	97	93
7	lngth, h2	cmpr, h6, id	h1, id	h4, h5	elng, sub, h3	add, id, h7	100	99	93

dnst = density, lngth = length, h = hidden unit, id = identity of transformed row, add = addition, sub = subtraction, elng = elongation, cmpr = compression. The horizontal lines within the table demarcate the layers of hidden units.

After recruiting a first hidden unit, network 8 substantially increases performance to 40 per cent correct on training problems and 44 per cent of test problems. A PCA of contributions at this point reveals the presence of three components – one continuing to deal with row density, and another with row length. The third, new component has loadings of decreasing size from the first hidden unit, and inputs concerning the identity of the transformed row, and the addition and elongation transformations. The non-linear power of the first hidden unit, plus consideration of these more conceptual inputs accounts for the large increase in performance. Knowing the identity of the transformed row is essential to correctly interpreting any additional information on the applied transformation.

To perform even better, network 8 next recruits a second hidden unit, allowing it to reach correct percentages of 66 on training and 58 on test problems. A PCA of contributions still yields three components, but their organization is different than previously. The first component has loadings from the four length inputs and the new hidden unit. The second component has loadings from hidden unit 1 and subtraction input. And the third component has loadings from identity of the transformed row, and the subtraction and elongation transformations. Density inputs have temporarily disappeared from consideration.

After recruiting a third hidden unit, this network further increases its performance to 73 and 76 per cent correct on training and test problems, respectively. It now has four components to its contributions. The first still deals with length inputs and hidden unit 2. The second component has loadings from the new hidden unit, the compression transformation, and a smaller contribution from density inputs. Component 3 has loadings from hidden unit 1, the subtraction transformation, and the identity of the transformed row. The fourth component has loadings from the second and third hidden units and the elongation transformation. The fact that length input continues to be accompanied by a hidden unit means that mere length is not used to decide which row has more, as was the case in the first two stages, before the network had at least two hidden units. Instead, length information is now being combined with other inputs in a nonlinear fashion to make decisions on relative number. At this stage, the network is considering inputs on only two of the four types of transformation, subtraction and elongation. So this network still has some distance to go in fully understanding conservation. Namely, it lacks sufficient computational power to process and coordinate other information on addition and compression transformations.

Not much has changed with the recruitment of hidden unit 4. PCA still reveals four components with loadings on the same factors as before, apart from some new, minor consideration of the addition transformation. Somehow the new hidden unit is not important enough to load on any of these four components. This explains why performance does not increase beyond the previous level. Percent correct is now 74 on training problems and 73 on test problems. All of the first four hidden units reside on the same first hidden-unit layer – they have no cascaded connections. As if sensing that further non-linearity is required, the algorithm recruits its next hidden unit on a different level, with cascaded connection weights from all of the four existing hidden units.

This results in a PCA with five components and a big jump in performance to 94 and 90 per cent correct on training and test problems, respectively. The first component looks the

same as before, length conditioned by hidden unit 2, but the rest of the components look substantially different. Component 2 deals with compression and hidden unit 3; component 3 with hidden units 4 and 5, and addition; component 4 with hidden unit 1 and the identity of the transformed row; and component 5 with elongation, subtraction, and hidden unit 3. Note that all existing hidden units are used, even the previously unused hidden unit 4. It is probably important that the new computational power afforded by hidden unit 5 enables an additional knowledge component allowing a greater spreading out of the various contributions. Spreading out these different functions is important to avoid the confusions caused by overloading of too much information on too few components.

Further representation spreading is enabled by recruitment of the sixth and seventh hidden units. The sixth hidden unit brings performance up to 99 and 97 per cent correct, respectively, on training and test problems. And the final, seventh hidden unit brings the network to nearly perfect performance on conservation problems. Notice the liberal use of information on the identity of the transformed row in three of the final six components. Identity is a key piece of information in many of the functions computed by the network in the final stage of learning. All of the important information on the conservation problem is well represented in these final solutions, although density information has been neglected since the second and last layer of hidden units began to form. As might be expected, in the strange environment where length is held constant in number-altering transformations, it is density inputs that are retained in network contributions and length inputs that are eventually ignored.

In effect, the network has progressed from responding based on how the rows look in terms of their length and density to reasoning about the effects of the transformation on the initial quantitative relation between the two rows. Such reasoning requires non-linear integration of information on the initial relation and the type and target of the transformation. For example adding an item has different results depending on both the row being added to and whether or not the two rows were initially equal in number.

In all of the conservation networks I have analyzed, whether constructed by the CC or SDCC algorithm, there is evidence of this progression from using the perceptual information of length and density to reasoning about the effects of particular transformations. This is the sort of shift that Piaget (1965) found with children. The knowledge-representation analyses presented here support Piaget's analysis, while specifying more fully, in computational terms, how such knowledge can be implemented and acquired within a plausible neural framework.

3.3.2.6 Summary of SDCC simulations

Like CC networks, SDCC networks covered all four principal conservation phenomena (acquisition, spurt, problem size, and length bias) and built knowledge representations mimicking the struggle between perceptual and cognitive solutions to the conservation task, a struggle eventually won in a comeback by cognitive strategies. At the same time, SDCC saved on numbers of network layers and connection weights. Past research, on a few non-psychological benchmark problems, found no reduction in connection weights with SDCC (Baluja and Fahlman 1994).

3.3.3 Conservation simulations with BP

Exactly the same training and testing scheme used with SDCC networks was tried with static BP networks. My original hypothesis was that, once finding a BP simulation that acquired conservation, knowledge-representation analysis might reveal that BP would not bother with perceptual solutions. Because these networks start with all the computational power they need, they might move directly to a more cognitive solution. As noted earlier, similar problems in capturing stages were reported for BP networks in the domain of integrating time, distance, and velocity cues for moving objects.

However, even when endowing BP networks with considerable computational power (one or two layers of 10 hidden units each), training them for extraordinary numbers of epochs (3000 to 12000 epochs), with systematic variation in key parameters of *epsilon* (learning rate, 0.1 to 0.5) and *mu* (momentum, 0.4 to 0.9), I have not been able to discover even a single case of successful acquisition of conservation. Best results were obtained with a large 13-10-10-2 topology, a low *epsilon* of 0.1, and a default *mu* of 0.9. Variation in training epochs did not matter much because there was no improvement after about 1000 epochs.

Fig. 3.5 shows results from a representative network, designed with a 13-10-10-2 topology. *Epsilon* was lowered to 0.1; *mu* was the default 0.9. The learning-rate parameter scales the sizes of weight changes and needs to be set to a moderate value to allow reasonably fast learning, but not so fast that weight changes oscillate wildly over more optimal values. With the *split-epsilon* technique used here, *epsilon* is divided by unit fan-in, which is 13 for the first hidden layer, 10 for the second hidden layer, and 10 for the outputs, all of which brings the effective *epsilon* values down to the vicinity used in the CC and SDCC conservation simulations. The momentum parameter provides weights with a relative degree of inertia, so that they will change less when the last change was small and change more when the last change was large. The idea is to induce larger weight changes when the weight is far from the error minimum, and smaller weight changes when the weight is approaching the error minimum. This network was trained for 3000 epochs. The results in the figure were recorded at 60 equally-spaced epochs. Some learning does occur here in early epochs, but performance never goes much above 50 per cent correct. The BP algorithm was modified here to stop learning when all output activations are within score-threshold of their targets for all of the training patterns, just as in CC and SDCC, so it did not stop until the 3000-epoch limit.

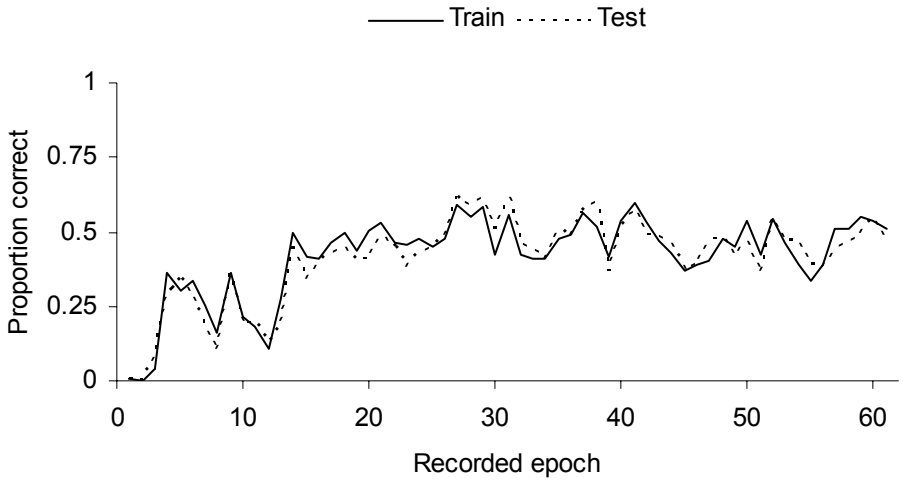

Fig 3.5 Performance on conservation problems by a 13-10-10-2 BP network over 3000 epochs.

Fig. 3.6 shows a different pattern that occurs often in these tests, also from a 13-10-10-2 BP network, but trained for 12000 epochs. Again, *epsilon* was 0.1 and *mu* was 0.9, the optimal values. Per-pattern error values were recorded at 60 equally-spaced epochs. Because this network shows no change in error from the second recorded epoch, it must be stuck in a local error minimum.

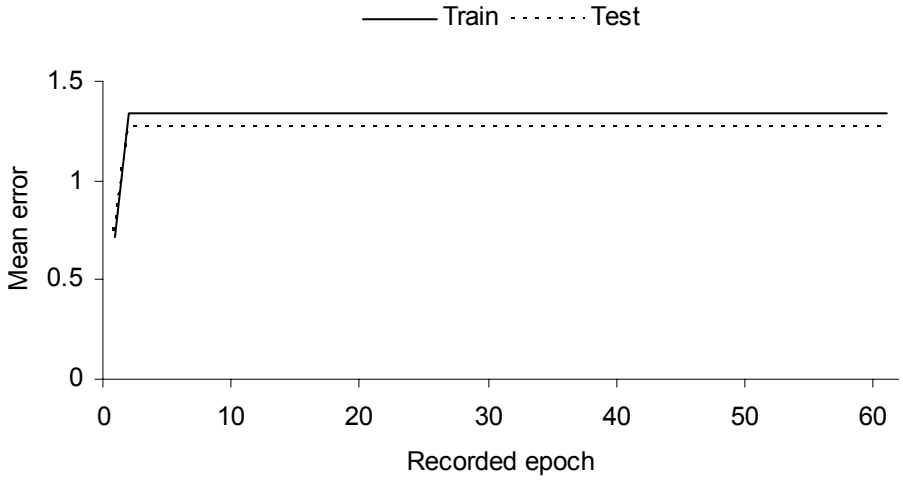

Fig 3.6 Performance on conservation problems by a 13-10-10-2 BP network over 12000 epochs.

With a very large number of possible designs for BP networks and a wide range of parameter settings to try, it is difficult to demonstrate that a particular learning algorithm cannot succeed, even on a difficult task like conservation acquisition. However, these results suggest that it will not be easy to get BP networks to acquire conservation as defined here.

3.4 Summary of comparison between static and constructive learning

In each of the four developmental domains where both static and constructive network algorithms have been applied, there is a clear superiority of constructive learning. Constructive algorithms, whether CC or SDCC, covered all of the basic psychological regularities in a natural and principled manner and generated sensible knowledge representations. In contrast, static BP networks suffer from any of three problems: they miss stages, fail to capture key psychological phenomena, or do not learn the training patterns.

BP networks missed important stages in the domain of integrating velocity, time, and distance cues for moving objects. They were either too underpowered to reach the terminal multiplicative stages, or else by virtue of starting with just one more unit they were too overpowered to capture intermediate additive stages. Network growth allowed constructive algorithms to capture all three levels of stages in this domain. Static BP networks also missed either stages 1 or 4 in the balance-scale domain. By changing biases in the training patterns, BP networks can be prodded to capture either stage 1 or stage 4, but they seem incapable of capturing both of these balance-scale stages. Moving from one region of connection-weight space to another can be difficult for a learning algorithm that cannot change its topology. The inability of BP networks to capture a full range of stages in these two domains points to a key reason for the success of CC networks in these domains. The gradual increase in computational power due to recruitment of hidden units allows CC networks to first perform at a more primitive level and then progress to more advanced stages, thus capturing the stages seen in children. Moving from a particular region of connection-weight space in stage 1 to a very different region in stage 4 of the balance scale is not a problem for CC networks because recruitment of hidden units effectively increases the dimensionality of that space, allowing greater flexibility in traversing the space.

In the domain of early category learning, BP networks did not capture any of the key phenomena. They could learn about the stimuli in the sense of reducing error, but they did not capture the results of either younger or older infants, showing neither a consistent similarity effect nor any correlation effect at all. Both of these effects, and the developmental transition between them, are captured by a network that is capable of growing while learning. Starting with little computational power requires CC networks to begin solving a relatively simple problem, discovery of stimulus features. Once the features have been noticed, then recruitment of additional computational power allows CC networks to track correlations among features.

Static BP networks have so far not mastered the conservation problem, despite designing networks that should have had sufficient computational power and time to learn and varying key learning parameters. The reasons for learning difficulty can be challenging to identify. But I would suggest that BP suffers from not being able to start from a simpler, under-powered perspective, a conjecture that is mindful of the benefits of starting small (Elman 1993). Both CC and SDCC start small in a topological and computational-power sense that does not require staging of the learning environment or limiting the window of processing, as Elman implemented with grammar learning. In constructive learning, starting with a small, underpowered network ensures that simple

ideas are acquired first and progressively more complex ideas are built on top of simpler ones.

Another advantage of constructive learning is that networks can more easily move off from local error minima in their continuing search for more global error minima in a high-dimensional weight space. Recruiting a new hidden unit effectively adds another dimension to weight space enabling a network to find a path to lower error.

These and other simulations make it clear that the superiority of CC over BP in covering developmental phenomena is not merely a matter of CC discovering more advantageous network topologies than the human designers of static networks were able to achieve. First it is noteworthy that CC and especially SDCC networks automatically construct a wide variety of network topologies. It is unlikely that all of these topologies are somehow fortuitously and exclusively optimal for every algorithm. Recall also that our BP simulations of the infant category-learning experiments used a modal CC network topology, in some conditions complete with cross connections that bypass hidden layers as in CC networks (Shultz and Cohen 2004). In none of these simulations did a BP network cover the basic characteristics of either younger or older infants, nor the transition from younger to older, despite having a topology identical to that of the final stage of successful CC networks.

We have also studied this issue of network topology in a non-developmental context. CC networks were trained on exclusive-or, parity, encoding, symmetry, binary addition, and negation problems. After training, weights in the CC networks were reset to small random values and each resulting network was trained with the BP algorithm. These pairs of CC and BP networks were compared in terms of whether they could learn these tasks, how fast they learned, and how well they generalized to novel examples. Their knowledge representations were also assessed. In contrast to their BP yoked controls, CC networks always learned the task with the default learning parameters, usually learned faster, and learned with less computation. These are relatively easy tasks, designed to exercise the BP algorithm. It is well known that BP can solve them, but it is interesting that BP networks do not always solve them. Thus, discovery of magically optimal network topologies is not the key to explaining the differences between static and constructive learning. The key difference seems to be a simultaneous search of connection-weight space (to find the right weight values) and topology space (to find the right network structure). Because topology space is searched during the learning of connection weights and progresses from simple to complex, learning is facilitated and psychological stages of acquisition are simulated.

3.5 Conclusions

In addition to the neurological and computational advantages of constructive learning, the present modeling shows advantages for capturing psychological phenomena in several domains of cognitive development. By starting small and building new conceptualizations on top of old ones, the CC and SDCC algorithms cover data that elude static BP learning. In some cases, BP was unable to learn, making data coverage impossible. In other cases, even if BP could learn the problem, it was unable to cover stages and stage sequences.

Another potential advantage of CC and SDCC over static learners is their essential continuity with knowledge-based algorithms. Knowledge-based cascade-correlation

(KBCC) recruits previously learned subnetworks as well as single hidden units, providing the capacity for existing knowledge to either facilitate or interfere with new learning (Shultz and Rivest 2001). It is difficult to see how static BP networks could find and use existing knowledge in so natural a fashion.

All of this is not to suggest that the BP algorithm has no place in developmental simulations. BP has probably generated more successful simulations in psychology and development than any other algorithm has and this success speaks for itself (Elman *et al*. 1996). It is also noteworthy that simple recurrent networks trained on small, finite-state grammars with back-propagation of error encode progressively longer temporal contexts as learning continues (Servan-Schreiber *et al*. 1991). This shows that recurrently-connected static networks can sometimes learn simple functions before complex ones. The present bakeoff simulations show that constructive learning does a better job than static algorithms at covering certain developmental phenomena involving stage sequences and particularly difficult learning.

Comparison of SDCC to CC in the domain of conservation acquisition suggests equivalent performance in all aspects except that SDCC requires fewer, and more realistic, levels of hidden units and fewer connection weights. Such results point to the promising nature of SDCC while not invalidating interpretations of the many earlier simulations using CC. The results also suggest that the details of network topology may not be as important as the increasing computational power of the network. The fact that SDCC creates such a wide variety of network topologies expressing the same functionality supports this conjecture.

Acknowledgements

This work is supported by a grant from the Natural Sciences and Engineering Research Council of Canada. This paper benefited from comments by Yoshio Takane, J-P Thivierge, Frédéric Dandurand, several participants attending the Attention and Performance meeting, and an anonymous reviewer.

References

Baluja S and Fahlman SE (1994). Reducing network depth in the cascade-correlation learning architecture. *Technical Report CMU-CS-94-209*, School of Computer Science, Carnegie Mellon University.

Baum BE (1989). A proposal for more powerful learning algorithms. *Neural Computation,* **1**, 201-207.

Buckingham D and Shultz TR (1996). Computational power and realistic cognitive development. In: *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society,* pp. 507-511. Mahwah, NJ: Erlbaum.

Buckingham D and Shultz TR (2000). The developmental course of distance, time, and velocity concepts: A generative connectionist model. *Journal of Cognition and Development,* **1**, 305-345.

Cohen LB and Arthur AE (2004). *The role of habituation in 10-month-olds' categorization*. Unpublished paper, Department of Psychology, University of Texas.

Elman JL (1993). Learning and development in neural networks: The importance of starting small. *Cognition,* **48**, 71-99.

Elman JL, Bates EA, Johnson MH, Karmiloff-Smith A, Parisi D and Plunkett K (1996). *Rethinking innateness: A connectionist perspective on development*. Cambridge MA: MIT Press.

Fahlman SE and Lebiere C (1990). The cascade-correlation learning architecture. In: DS Touretzky, ed. *Advances in neural information processing systems 2*, pp. 524-532. Morgan Kaufmann, Los Altos, CA.

Fodor JA (1980). Fixation of belief and concept acquisition. In M. Piatelli-Palmarini, ed. *Language and learning: The debate between Chomsky and Piaget*, pp. 143-149. Cambridge, MA: Harvard Press.

Gould E, Tanapat P, Hastings NB and Shors TJ (1999). Neurogenesis in adulthood: A possible role in learning. *Trends in Cognitive Sciences,* **3**, 186-192.

Kempermann G and Gage FH (1999). New nerve cells for the adult brain. *Scientific American,* **280**, 48-53.

Mareschal D and Shultz TR (1996). Generative connectionist networks and constructivist cognitive development. *Cognitive Development,* **11**, 571-603.

McClelland JL and Jenkins E (1991). Nature, nurture, and connections: Implications of connectionist models for cognitive development. In K van Lehn, ed. *Architectures for intelligence: The twenty-second (1988) Carnegie symposium on cognition*, pp. 41-73. Hillsdale, NJ: Erlbaum.

Miller PH, Grabowski TL and Heldmeyer KH (1973). The role of stimulus dimensions in the conservation of substance. *Child Development,* **44**, 646-650.

Munakata Y and McClelland JL (2003). Connectionist models of development. *Developmental Science,* **6**, 413–429.

Piaget J (1965). *The child's conception of number*. New York: Norton.

Purves D, White LE and Riddle DR (1996). Is neural development Darwinian? *Trends in Neuroscience,* **19**, 460-464.

Quartz SR (1993). Neural networks, nativism, and the plausibility of constructivism. *Cognition,* **48**, 223-242.

Quartz SR (2003). Learning and brain development: A neural constructivist perspective. In PT Quinlan, ed. *Connectionist models of development: Developmental processes in real and artificial neural networks*, pp. 279-309. New York: Psychology Press.

Quartz SR and Sejnowski TJ (1997). The neural basis of cognitive development: A constructivist manifesto. *Behavioural and Brain Sciences,* **20**, 537-596.

Servan-Schreiber D, Cleeremans A and McClelland JL (1991). Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, **7**, 161-193.

Shultz TR (1998). A computational analysis of conservation. *Developmental Science,* **1**, 103-126.

Shultz TR (2003). *Computational developmental psychology*. Cambridge, MA: MIT Press,.

Shultz TR and Cohen LB (2004). Modeling age differences in infant category learning. *Infancy,* **5**, 153-171.

Shultz TR Mareschal D and Schmidt WC (1994). Modeling cognitive development on balance scale phenomena. *Machine Learning,* **16**, 57-86.

Shultz TR, Mysore SP and Quartz SR (2006, in press). Why let networks grow? In: D Mareschal, S Sirois and G Westermann, eds. *Constructing cognition: Perspectives and prospects*. Oxford: Oxford University Press.

Shultz TR, Oshima-Takane Y and Takane Y (1995). Analysis of unstandardized contributions in cross connected networks. In: D Touretzky, G Tesauro and TK Leen, eds. *Advances in neural information processing systems 7*, pp. 601-608. Cambridge, MA: MIT Press.

Shultz TR and Rivest F (2001). Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science,* **13**, 43-72.

Siegler RS (1976). Three aspects of cognitive development. *Cognitive Psychology,* **8**, 481-520.

Wilkening F (1981). Integrating velocity, time, and distance information: A developmental study. *Cognitive Psychology,* **13**, 231-247.

Younger BA and Cohen LB (1983). Infant perception of correlations among attributes. *Child Development,* **54**, 858-867.

Younger BA and Cohen LB (1986). Developmental change in infants' perception of correlations among attributes. *Child Development,* **57**, 803-815.