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

Running head: KNOWLEDGE-BASED CASCADE-CORRELATION

Knowledge-based Cascade-correlation: Using Knowledge to Speed Learning

Thomas R. Shultz
Department of Psychology
McGill University

Francois Rivest
Department of Computer Science
McGill University

Thomas R. Shultz, Professor
Department of Psychology
McGill University, 1205 Penfield Ave.
Montreal, Quebec, Canada H3A 1B1.
E-mail: thomas.shultz@mcgill.ca
Phone: 514 398-6139
Fax: 514 398-4896

Keywords: knowledge, learning, cascade-correlation, generative networks

Abstract

Research with neural networks typically ignores the role of knowledge in learning by initializing the network with random connection weights. We examine a new extension of a well-known generative algorithm, cascade-correlation. Ordinary cascade-correlation constructs its own network topology by recruiting new hidden units as needed to reduce network error. The extended algorithm, knowledge-based cascade-correlation (KBCC), recruits previously learned sub-networks as well as single hidden units. This paper describes KBCC and assesses its performance on a series of small, but clear problems involving discrimination between two classes. The target class is distributed as a simple geometric figure. Relevant source knowledge consists of various linear transformations of the target distribution. KBCC is observed to find, adapt, and use its relevant knowledge to significantly speed learning.

1. Existing Knowledge and New Learning

Learning in neural networks is typically done "from scratch", without the influence of previous knowledge. However, it is clear that people make extensive use of their existing knowledge in learning (Heit 1994; Keil 1987; Murphy 1993; Nakamura 1985; Pazzani 1991; Wisniewski 1995). Use of knowledge is likely responsible for the ease and speed with which people are able to learn new material, although interesting interference of knowledge with learning can also occur. Neural networks fail to use knowledge in new learning because they begin learning from initially random connection weights.

Here we examine a connectionist algorithm that uses its existing knowledge to learn new problems. This algorithm is an extension of cascade-correlation (CC), a generative learning algorithm that has proved to be useful in the simulation of cognitive development (Buckingham & Shultz, 1994; Mareschal & Shultz, 1999; Shultz 1998; Shultz, Buckingham, & Oshima-Takane, 1994; Shultz, Mareschal, & Schmidt, 1994; Sirois & Shultz, 1998). Ordinary CC creates a network topology by recruiting new hidden units into a feed-forward network as needed in order to reduce error (Fahlman & Lebiere, 1990). The extended algorithm, called knowledge-based cascade-correlation (KBCC), recruits whole sub-networks that it has already learned, in addition to the untrained hidden units recruited by CC (Shultz & Rivest, 2000a). The extended algorithm thus adapts old knowledge in the service of new learning. KBCC trains connection weights to the inputs of its existing sub-networks to determine whether their outputs correlate well with the network's error on the problem it is currently learning. Consistent with the conventional terminology of the literatures on analogy and transfer of learning, we refer to these existing sub-networks as *source* knowledge and to the current learning task as a *target* problem. These previously learned source networks compete with each other and with conventional untrained candidate hidden units to be recruited into the target network learning the current problem. As we discuss later, KBCC is similar in spirit to recent neural network research on transfer of knowledge, multitask learning, sequential learning, lifelong learning, input re-coding, knowledge insertion, and modularity, but it incorporates these ideas by learning, storing, and searching for knowledge within a generative network approach.

We first describe the KBCC algorithm, show its learning speed performance on a number of learning problems that could potentially benefit from prior knowledge, and then discuss its advantages and limitations in the context of the current literature on knowledge and learning in neural networks.

2. Description of KBCC

2.1 Overview

Because KBCC is an extension of CC, it uses many of CC's ideas and mathematics. As we describe KBCC, we note particular differences between the two algorithms. Both algorithms specify learning in feed-forward networks, adjust weights based on training examples presented in batch mode, and operate in two phases: output phase and input phase. In output phases, connection weights going into output units are adjusted in order to reduce error at the output units. In input phases, the input weights going into recruitment candidates are adjusted in order to maximize a modified correlation between activation of the candidate and error at the output units. Networks in both algorithms begin with only input and output units. During learning, networks alternate between input and output phases, respectively, depending on whether a new

candidate is being recruited or not. We begin with the contrasting features of networks and units, and proceed to discuss the output phase, input phase, and connection scheme.

2.2 Networks and Units

The major new idea in KBCC is to treat previously learned networks just like candidate hidden units, in that they are all candidates for recruitment into a target network. A sample KBCC network with two input units and a bias unit is pictured in Figure 1. This particular network has two hidden units, the first of which is a sub-network and the second of which is a single unit. Later hidden units are installed downstream of existing hidden units.

===Insert Figure 1 about here===

A single unit and a network both describe a differentiable function, which is what is required for learning in most feed-forward learning algorithms. In the case of a single unit, such as hidden unit 2 in Figure 1, this is a function of one variable, the net input to the unit. Net input to a unit *i* is the weighted sum of its input from other units, computed as:

$$x_i = \sum_j w_{ij} a_j \qquad (1)$$

where *i* indexes the receiving unit, *j* indexes the sending units, *a* is the activation of each sending unit, and *w* is the connection weight between units *j* and *i*.

The function for a single unit is often the centered logistic sigmoid function, in the range of -0.5 to 0.5:

$$f(x) = \frac{1}{1+e^{-x}} - 0.5 \qquad (2)$$

where *x* is the net input to the unit. Other activation functions used in CC and KBCC are the *asigmoid* (logistic sigmoid) and *Gaussian* functions, both in the range 0.0 to 1.0.

In the case of an existing sub-network, things are a bit more complicated because, unlike a single unit, there may be multiple inputs from each upstream unit and multiple outputs to each downstream unit, as illustrated by hidden unit 1 in Figure 1. For each such sub-network, the input weights and the output weights are each represented as a vector of vectors. Because the internal structure of a previously trained network is known, it can be differentiated in order to compute the slopes needed in weight adjustment, just as is commonly done with single hidden units.

In KBCC, there is a list of current candidate networks, referred to as a parameter called *CandidateNetworksList*.

We refer to the weights entering candidate hidden units and candidate networks as input-side weights. The input-side weights are trained during input phases as explained in section 2.5.

2.3 Notation

Before presenting the algorithm in detail, it is helpful to describe the notation used in the various equations.

$w_{o_u,o}$: Weight between output o_u of unit[1] u and output unit o.

w_{o_u,i_c}: Weight between output o_u of unit u and input i_c of candidate c.

$f'_{o,p}$: Derivative of the activation function of output unit o for pattern p.

$\nabla_{i_c} f_{o_c,p}$: Partial derivative of candidate c output o_c with respect to its input i_c for pattern p.

$V_{o,p}$: Activation of output unit o for pattern p.

$V_{o_c,p}$: Activation of output o_c of candidate c for pattern p.

$V_{o_u,p}$: Activation of output o_u of unit u for pattern p.

$T_{o,p}$: Target value of output o for pattern p.

2.4 Output Phase

In the output phase, all weights entering the output units, called output weights, are trained in order to reduce error. As in CC networks, KBCC networks begin and end their learning career in output phase. The weights that fully connect the network at the start of training are initialized randomly using a uniform distribution with range [-*WeightsRange*, *WeightsRange*]. The default value is *WeightsRange* = 1.0. A bias unit, with an activation of 1.0, feeds all hidden and output units in the network.

The output weights are trained using the *quickprop* algorithm (Fahlman 1988). The quickprop algorithm is significantly faster than standard back-propagation because it supplements the use of slopes with second-order information on curvature, which it estimates with the aid of slopes on the previous step. Quickprop has parameters for learning rate ε, maximum growth factor μ, and weight decay γ. The learning rate parameter controls the amount of linear gradient descent used in updating output weights. The maximum growth factor constrains the amount of weight change. The amount of decay times the current weight is added to the slope at start of each output phase epoch.[2] This keeps weights from growing too large. The default values for these three parameters are $\varepsilon = 0.175 / n$, $\mu = 2.0$, and $\gamma = 0.0002$, respectively, where n is the number of patterns.

The function to minimize in the output phase is the sum-squared error over all outputs and all training patterns:

$$F = \sum_o \sum_p \left(V_{o,p} - T_{o,p}\right)^2 \qquad (3)$$

The partial derivative of F with respect to the weight $w_{o_u,o}$ is given by

$$\frac{\partial F}{\partial w_{o_u,o}} = 2\sum_p \left(V_{o,p} - T_{o,p}\right) f'_{o,p} V_{o_u,p} \qquad (4)$$

[1] We use *unit* to refer to any of the bias, input, or hidden units except when otherwise stated. Hidden units include both single units and sub-networks.

[2] An epoch is a batch presentation of all of the traning patterns.

The activation function for output units is generally the sigmoid function shown in Equation 2. Linear activation functions can also be used for output units. When the sigmoid function is used for output units, a small offset is added to its derivative to avoid getting stuck at the flat points when the derivative goes to 0 (Fahlman 1988). By default, this *SigmoidOutputPrimeOffset* = 0.1.

An output phase continues until any of following criteria is satisfied:

1. When a certain number of epochs pass without solution, there is a shift to the input phase. By default this number of epochs *MaxOutputEpoch* = 100.
2. When error reduction stagnates for few consecutive epochs, there is a shift to the input phase. Error is measured as in Equation 3, and must change by at least a particular proportion of its current value to avoid stagnation. By default, this proportion, called *OutputChangeThreshold*, is 0.01. The number of consecutive output phase epochs over which stagnation is measured is called *OutputPatience* and is 8 by default.
3. When all output activations are within some range of their target value, that is, when $|V_{o,p} - T_{o,p}| \leq$ *ScoreThreshold* for all *o* outputs and *p* patterns, victory is declared and learning ceases. By default, *ScoreThreshold* = 0.4, which is generally considered appropriate for units with sigmoid activation functions (Fahlman 1988). The *ScoreThreshold* for output units with linear activation functions would need to be set at the level of precision required in matching target output values.

2.5 Input Phase

In the input phase, a new hidden unit is recruited into the network. This new unit is selected from a pool of candidates. The candidates receive input from all existing network units, except output units, and these input weights are trained by trying to maximize the correlation between activation on the candidate and network error. During this training, all other weights in the network are frozen. The candidate that gets recruited is the one that is best at tracking the network's current error. In KBCC, candidates include not only single units as in CC, but also networks acquired in past learning.

N is the *NumberCandidatesPerType*, which is 4 by default. Weights entering *N* single-unit candidates are initialized randomly using a uniform distribution with range [-*WeightsRange*, *WeightsRange*] as in the output phase. Again, the default value is *WeightsRange* = 1.0. For each network in the *CandidateNetworksList*, input weights for *N*-1 instances are also initialized. Each input-side connection of these units is initialized using the same scheme as for the basic network weights, with one exception. The exception is that one instance of each stored network has its weight matrix initialized with weights of 1.0 connecting corresponding inputs of target networks to source networks and weights of 0.0 elsewhere. This is to enable use of relevant exact knowledge without much additional training. We call this the *directly connected* version of the knowledge source. Activation functions of the single units are generally all sigmoid, asigmoid, or Gaussian, with sigmoid being the default.

As in output phases, all of these input-side weights are trained with quickprop (with $\varepsilon = 1.0/nh$,[3] where n is the number of patterns and h is the number of units feeding the candidates, $\mu = 2.0$, and $\gamma = 0.0000$). The function to maximize is the average covariance of the activation of each candidate (independently) with the error at each output, normalized by the sum-squared error. For candidate c, the function is given by

$$G_c = \frac{\sum_{o_c}\sum_{o}\left|\sum_{p}\left(V_{o_c,p} - \overline{V}_{o_c}\right)\left(E_{o,p} - \overline{E}_{o}\right)\right|}{\#O_c \cdot \#O \cdot \sum_{o}\sum_{p} E_{o,p}^2} \tag{5}$$

where $\overline{E}_o$ is the mean error at output unit o, and $\overline{V}_{o_c}$ is the mean activation output o_c of candidate c.

The output error at pattern p is

$$E_{o,p} = \begin{cases} \left(V_{o,p} - T_{o,p}\right) & \text{if} \quad RawError = true \\ \left(V_{o,p} - T_{o,p}\right) f'_{o,p} & otherwise \end{cases} \tag{6}$$

G_c is standardized by both the number of outputs for the candidate c ($\#O_c$) and the number of outputs in the main network ($\#O$). By default, *RawError = false*.[4]

The partial derivative of G_c with respect to the weight w_{o_u,i_c} between output o_u of unit u and input i_c of candidate c is given by

$$\frac{\partial G_c}{\partial w_{o_u,i_c}} = \frac{\sum_{o_c}\sum_{o}\sum_{p}\sigma_{o_c,o}\left(E_{o,p} - \overline{E}_o\right)\nabla_{i_c} f_{o_c,p} V_{o_c,p}}{\#O_c \cdot \#O \cdot \sum_{o}\sum_{p} E_{o,p}^2} \tag{7}$$

where $\sigma_{o_c,o}$ is the sign of the covariance between the output o_c of candidate c and the activation of output unit o.

An input phase continues until either of following criteria is met:

1. When a certain number of input phase epochs passes without solution, there is a shift to output phase. By default this *MaxInputEpoch* = 100.
2. When at least one correlation reaches a *MinimalCorrelation* (default value = 0.2) and correlation maximization stagnates for few consecutive input phase epochs, there is a shift to output phase. Correlation is measured as in Equation 5, and must change by at least a particular proportion of its current value to avoid stagnation. By default, this proportion, called *InputChangeThreshold*, is 0.03. The number of consecutive input phase epochs over which correlation stagnation is measured is called *InputPatience* and is 8 by default.

---

[3] The *1/n* (output phase) and *1/nh* (input phase) fraction cannot be described as part of the objective function of their respective phases as in standard back-propagation because ε is not used in the quadratic estimation of the curve in quickprop. It is a heuristic from Fahlman (1988) to set ε dynamically.

[4] The variation of the error function $E_{o,p,}$ which depends on *RawError*, comes from Fahlman's (1991) CC code.

When a criterion for shifting to output phase is reached, a set of weights is added from the outputs of the best candidate to each output of the network. All other candidate units are discarded, and the newly created weights are initialized with small random values (between 0.0 and 1.0), with the sign opposite to that in correlation.

2.6 Connection Scheme in KBCC

Figure 2 shows the connection scheme for a sample KBCC network with two inputs, two outputs, one recruited network, and a recruited hidden unit. The recruited network, labeled H1 because it was the first recruited hidden unit, has two input units, two output units, and a single hidden unit, each labeled with a prime (') suffix. The main network and the recruited sub-network each have their own bias unit. Figure 2 reveals that the recruited sub-network is treated by the main network as a computationally encapsulated module, receiving input from the inputs and bias of the main network and sending output to later hidden units and the output units. Other than that, the main network has no interaction with the work of the sub-network.

===Insert Figure 2 about here===

3. Applications of KBCC

To evaluate the behavior of KBCC, we applied it to learning in two different paradigms. One paradigm tests whether KBCC can find and use its relevant knowledge in the solution of a new problem and whether this relevant knowledge shortens the time it takes to learn the new problem. A second paradigm tests whether KBCC can find and combine knowledge of components to learn a new, more complex problem comprised of these components, and whether use of these knowledge components speeds learning. In each paradigm there are two phases, one in which source knowledge is acquired and a second in which this source knowledge might be recruited to learn a target problem. These experiments are conducted with toy problems with a well-defined structure so that we can clearly assess the behavior of the KBCC algorithm. In each problem, networks learn to identify whether a given pattern falls inside a class that has a two-dimensional uniform distribution. The networks have two linear inputs and one sigmoid output. The two inputs describe two real-valued features; the output indicates whether this point is inside or outside a class of a particular distribution with a given shape, size, and position. The input space is a square centered at the origin with sides of length 2. Target outputs specify that the output should be 0.5 if the point described in the input is inside the particular class and -0.5 if the point is not in the class. In geometric terms, points inside the target class fall within a particular geometric figure; points outside of the target class fall outside of this figure. Networks are trained with a set of 225 patterns forming a 15 x 15 grid covering the whole input space including the boundary. For each experiment, there are 200 randomly determined test patterns uniformly distributed over the input space. These are used to test generalization, and are never used in training. We used this task to facilitate design and description of problems, variation in knowledge relevance, and identification of network solutions (by comparing output plots to target shapes). These problems, although small and easy to visualize, are representative of a wide range of classifier and pattern recognition problems. Knowledge relevance involved differences in the position and shape of the distribution of patterns that fell within the designated class (or figure). Degree of relevance was indexed by variation in the amounts of translation, rotation, and scaling. So-called *irrelevant* source knowledge involved learning a class whose distribution has a different geometric shape than the target class.

We ran 20 KBCC networks in each condition of each experiment in order to assess the statistical reliability of results, with networks differing in initial output and input weights. Learning speed was measured by epochs to learn. Use of relevant knowledge was measured by identifying the source of knowledge that was recruited during input phases.

In these experiments, networks learning a target task have zero, one, or two source networks to draw upon in different conditions. In each input phase of single source experiments, there are always eight candidates, four of them being previously learned networks and four of them being single units. In control conditions without knowledge (no source networks), all eight candidates are single units. These control networks are essentially CC networks. In conditions with two source networks in memory, there are three candidates representing one source network, three candidates representing the other source network, and three single unit candidates. The reason for having multiple candidates for each unit and source network is to be able to provide a variety of initial input weights at the start of the input phase. This enables networks to try a variety of different mappings of the target task to existing knowledge.

4. Finding and Using Relevant Knowledge

We did two kinds of experiments to assess the impact of source knowledge on learning a target task. In one kind of experiment, we varied the relevance of the single source of knowledge the network possessed to determine whether KBCC would learn faster if it had source knowledge that was more relevant. In a second kind of experiment, we gave networks two sources of knowledge, varying in relevance to a new target problem, to discover whether KBCC would opt to use more relevant source knowledge. To assess the generality of our results, we conducted both types of experiments with three different sets of linear transformations of the input space: translation, size changes, and rotation. In all of these experiments, KBCC networks acquired source knowledge by learning one or two problems and then learned another, target problem for which the impact of the previously acquired source knowledge could be assessed. We first consider problems of translation, then sizing, and finally rotation.

4.1 Translation[5]

In translation problems, degree of knowledge relevance was varied by changing the position of two-dimensional geometric figures. The target figure in the second (or target) phase of knowledge-guided learning was a rectangle with width of 0.4 and height of 1.6 centered at (-4/7, 0) in the input space. For translation problems, we first consider the effects of single-source knowledge on learning speed in the target phase and then the knowledge that is selected when two sources of knowledge are available.

4.1.1 Effects of Single-source Knowledge Relevance on Learning Speed

In this experiment, networks had to learn a rectangle (named RectL) positioned a bit to the left of center in the input space, after having previously learned a rectangle, or two rectangles, or a circle at particular positions in the input space. The various experimental conditions are shown in Table I, in terms of the name of the condition, a description of the stimuli that had been previously learned in phase 1, and the relation of those stimuli to the target rectangle (RectL). Previous, phase-1 (or source) learning created the various knowledge conditions for the new learning of the target rectangle in the target phase. Conditions included exact knowledge (i.e., identical to the target), exact but overly complex knowledge, relevant

[5] A preliminary version of the translation results were presented in Shultz and Rivest (2000a).

knowledge that was either near to or far from the target, overly complex knowledge that was far from the target, and irrelevant knowledge. In a control condition, networks had no knowledge at all when beginning the target task, which is equivalent to ordinary CC.

===Insert Table I about here===

A factorial ANOVA of the epochs required to reach victory yielded a main effect of knowledge condition, $F(6, 133) = 33$, $p < .0001$. Mean epochs to victory in each condition, with standard deviation bars and homogeneous subsets, based on the *LSD* post hoc comparison method, are shown in Figure 3. Means within a homogeneous subset are not significantly different from each other. Figure 3 reveals that exact knowledge, whether alone or embedded in an overly complex structure produced the fastest learning, followed by relevant knowledge, distant and overly complex knowledge and irrelevant knowledge, and finally the control condition without any knowledge.

===Insert Figure 3 about here===

Some example output activation diagrams for one representative network from this simulation are shown in Figure 4. In these plots, white regions of the input space are classified as being inside the rectangle (network response > 0.1), black regions outside of the rectangle (network response < -0.1), and gray areas are uncertain, meaning that the network gives a borderline, unclassifiable response somewhere between -0.1 and 0.1. The 225 target training patterns, forming a 15 x 15 grid covering the whole input space, are shown as points in each plot. Figure 4a shows the source knowledge learned by this network in the exact but overly complex condition. The two white regions indicate the two rectangles constituting the target class for this condition. Such shapes are somewhat irregular, even if completely correct with respect to the training patterns, because they are produced by sampling the network on a fine grid of 220 x 220 input patterns.

===Insert Figure 4 about here===

Figure 4b shows this same KBCC system's output at the end of the first output phase of target training. There are no white regions in this plot because the network has learned to classify most of the input patterns as being outside of the target class; the network is, on our definition, unsure of patterns within the first column of the input space. Because only 33 of the 225 training patterns fall within the target class, this is the best that the target network can do without any hidden units. Such behavior was common for the first output phase of learning a small rectangular target in every condition of every experiment reported in this paper.

Figure 4c shows this network's final solution at the end of the second output phase of target training, after having recruited the source knowledge shown in Figure 4a. In this single-source experiment, exact but overly complex source knowledge was very effective in speeding up learning, as reported in Figure 3. Comparison of Figures 4a and 4c suggests that KBCC uses the recruited source network only when the input represented on the *x*-axis is less than about -3/14; otherwise it uses its direct input-to-output weights. Examination of the output weights from the bias unit, the *x*-axis input unit, and the recruited hidden network confirmed that this is the case. These weights are such that the recruited hidden network can raise the weighted sum feeding the output unit above 0.0 only in the region of the target rectangle. Notice how closely the shape of the final solution in Figure 4c resembles that of the exactly correct portion of the source network in Figure 4a.

Because one of the candidate source networks is initialized using direct input connections (i.e., weights that map input *i* to source network input *i* are set to 1.0 and all others to 0.0), KBCC always recruits that candidate source first in exact knowledge conditions. For 65% of these exact source networks, no further recruitment was necessary; for the remaining 35%, some additional recruitment was necessary to adjust to a few borderline patterns. A directly connected source candidate network was much less likely to be recruited in the close (25%) and far (0%) relevant conditions.

4.1.2 Selection of Relevant Knowledge from Two Sources

In this experiment, networks first learned two tasks of varying relevance to the target task of learning RectL, the rectangle placed slightly to the left of the origin. The names of the various knowledge conditions, their relations to the target, and the mean times each network was recruited during input phases of target learning are shown in Table II. Descriptions of the figures designated by each condition name were provided in Table 1. The two recruitment means in each row were compared with a *t*-test for paired samples, except in the Exact vs. Relevant condition, where there was no variation in either variable. In every other case, the mean difference was significant at $p < .001$, $df = 19$. The pattern of differences shows that target networks preferred exact knowledge, even when it was embedded in overly complex knowledge. They also preferred simple exact knowledge to overly complex knowledge that had exact knowledge embedded within it. Interestingly, a circle source was more often recruited than a source rectangle positioned at the right. Only two single-hidden-units were recruited by the 120 networks in this experiment.

===Insert Table II about here===

The results of this dual-source experiment make sense given our analysis of the single-source experiment. Exact source knowledge is preferred over inexact source knowledge because it makes a nearly perfect match when accessed with direct connections. Overly complex exact source knowledge is less apt to be recruited than is simple exact source knowledge because it correlates less well with error in the case of directly connected sources. Perhaps the circle source knowledge is recruited over the far source rectangle because the circle is closer to the target rectangle even though it is the wrong shape.

4.2 Sizing

In sizing problems, knowledge relevance was varied by changing the size of two-dimensional geometric figures. The target figure in the second phase of knowledge-guided learning was a rectangle as were the figures in several of the knowledge conditions. Rectangles were always centered at (0, 0) in the input space and always had a height of 22/14.

4.2.1 Effects of Single-source Knowledge Relevance on Learning Speed

In this experiment, several knowledge conditions varied the width of the first-learned rectangle. Because scaling the width up and scaling the width down do not produce the same results, we included conditions with either small or large target rectangles. The various conditions, which also included irrelevant knowledge in the form of a circle and no knowledge at all, are shown in Table III.

===Insert Table III about here===

A factorial ANOVA of the epochs to victory when the small rectangle was the target yielded a main effect of knowledge condition, $F(4, 95) = 103$, $p < .0001$. The mean epochs to victory, with standard deviation bars and homogeneous subsets, based on the *LSD* post hoc comparison method, are shown in Figure 5. Relevant knowledge, regardless of distance from the target, produced faster learning than did irrelevant knowledge and no knowledge. This suggests that scaling down in size is not much affected by the amount of scaling required. The relatively few epochs required in phase 2 indicates that scaling down in size is relatively easy for these networks to learn.

===Insert Figure 5 about here===

A factorial ANOVA of the epochs to victory when the large rectangle was the target also yielded a main effect of knowledge condition, $F(4, 95) = 74$, $p < .0001$, but with a somewhat different pattern of results. The mean epochs to victory, with standard deviation bars and homogeneous subsets, based on the *LSD* post hoc comparison method, are shown in Figure 6. Exact knowledge yielded the fastest learning, followed in turn by near relevant knowledge, far relevant knowledge, and finally by no knowledge and irrelevant knowledge. This means that scaling up in size gets more difficult with the amount of scaling required. In this case, irrelevant knowledge did not speed up learning, as compared to the no-knowledge control. Examination of phase-1 source acquisition results confirmed that small rectangles were easier to learn than large rectangles, in terms of both hidden units recruited and epochs to learn.

===Insert Figure 6 about here===

When learning a small target rectangle, the percent of networks recruiting a directly connected source network decreased from 100% in the exact source knowledge condition to 80% in the near relevant source knowledge condition to 45% in the far relevant source knowledge condition. Figures 7 and 8 present output activation diagrams for networks learning a small target rectangle, recruiting either near relevant or far relevant directly connected source knowledge, respectively. Again, in both cases, there is a striking resemblance between the shape of the source knowledge and that of the final solution. As with translation experiments, the networks here learn to classify all patterns as being outside of the target class during the first output phase.

===Insert Figures 7 and 8 about here===

However, when learning a large target rectangle, networks do the opposite; that is, they learn to classify all patterns as being inside of the target class during the first output phase. This is because many more of the training patterns (121 of 225) fall within the target class when the target is a large rectangle. The percent of networks recruiting the directly connected source network when learning a large target rectangle was 100% in the exact and near source knowledge conditions and 75% in the far relevant source knowledge condition. Figures 9 and 10 show output activation diagrams for networks learning a large target rectangle, with either near relevant or far relevant source knowledge, respectively.

===Insert Figures 9 and 10 about here===

Any lingering error during target learning, whether scaling down to a small rectangle or scaling up to a large rectangle, involves patterns near the four corners of the target rectangle, such corners being regions of intersecting hyper-planes being learned by the network. When scaling down to learn a small target rectangle, network recruitment sharpens these corners,

making target learning rather fast. In contrast, when scaling up to a large target rectangle, network recruitment smoothes these corners, thus prolonging target learning in order to re-sharpen the corners. When scaling up to a large rectangle, the amount of corner smoothing and eventual re-sharpening grows with the degree of scaling. Because no additional sharpening of corners is required when scaling down to a small rectangle, learning speed is rather fast and does not vary with the degree of scaling required.

Mean input connection weights for the critical *x*-axis input units learned in the sizing experiment while recruiting a directly connected source network are plotted in Figure 11. Because recruiting exact knowledge requires no rescaling of inputs, these connection weights are about 1, regardless of the size of the target rectangle. With a small target rectangle, these weights increase with the amount of scaling required; with a large target rectangle, these weights decrease with the amount of scaling required. Such trends make sense because when scaling down to a small target rectangle, the inputs to the small target would need to be scaled up with larger weights in order to effectively use the larger source network. In contrast, when scaling up to a large target rectangle, the inputs to the large target would need to be scaled down with smaller weights in order to effectively use the smaller source network. During these recruitments, connection weights for *y*-axis input units were always about 1 because rectangle height was constant, and all other connection weights were about 0 because they were unimportant.

===Insert Figure 11 about here===

4.2.2 Selection of Relevant Knowledge from Two Sources

In this experiment, networks first learned two tasks of varying relevance to the target task. The names of the various knowledge conditions, their relations to the target, and the mean times each network was recruited during input phases are shown in Table IV. The descriptions of the figures associated with each condition name were provided in Table III. The two means in each row were compared with a *t*-test for paired samples. Results are shown in the last two columns of Table IV. Exact knowledge was preferred over relevant or irrelevant knowledge. Relevant knowledge was preferred over irrelevant knowledge only when scaling down to a smaller rectangle. The large number of recruited networks in the relevant vs. irrelevant, scaling-up condition reflects the relative difficulty of learning in this condition. Again, the longer that learning continues the more recruitment is required. In this experiment, two of the 120 networks each recruited 1 single-hidden-unit, and two others recruited 2 single-hidden-units, for a total of 6. All six of these hidden units recruited were all in the scaling-up conditions of the experiment.

===Insert Table IV about here===

The results of this dual-source sizing experiment make sense given our analysis of the single-source sizing experiment. Exact source knowledge is preferred over inexact source knowledge because it makes a nearly perfect match when accessed with direct connections. When scaling down to a small rectangle, relevant inexact source knowledge is preferred to irrelevant source knowledge because the recruitment sharpens the critical corners of the target figure, which is rectangular like the relevant sources. In contrast, when scaling up to a large rectangle, there is no advantage for relevant source knowledge because recruiting smoothes the critical target corners thus requiring additional re-sharpening through further learning.

4.3 Rotation

In rotation problems, knowledge relevance was varied by rotating a rectangle of size 1.5 x 0.5 or a cross comprising two such rectangles that were offset 90 degrees. All figures were centered at (0, 0). The target figure in the second phase of knowledge-guided learning was a vertically oriented rectangle (Rect90).

4.3.1 Effects of Single-source Knowledge Relevance on Learning Speed

In this experiment, networks had to learn a vertical rectangle (Rect90) after having previously learned a rectangle or a cross or a circle. The various experimental conditions are shown in Table V.

===Insert Table V about here===

A factorial ANOVA of the epochs to victory yielded a main effect of knowledge condition, $F(6, 133) = 20$, $p < .0001$. The mean epochs to victory, with standard deviation bars and homogeneous subsets, based on the *LSD* post hoc comparison method, are shown in Figure 12. Exact knowledge produced the fastest learning, followed by exact knowledge embedded in an overly complex structure and irrelevant knowledge, near and distant relevant knowledge, distant and overly complex knowledge, and finally the control condition without any knowledge.

===Insert Figure 12 about here===

Figure 13 presents output activation diagrams for a network learning a vertical rectangle after recruiting directly connected, exact but overly complex source knowledge. The target solution is accomplished by gradually shrinking the horizontal arms of the recruited cross via more recruiting and adjusting of direct input-to-output weights. The top and bottom of the vertical portion of the recruited cross (Figure 13a) resemble those in the emerging (Figure 13b) and final target solutions (Figure 13c). The directly connected version of the source knowledge was recruited by 100% of networks in the exact knowledge condition and by 65% of networks in the exact but overly complex condition. This explains why these two conditions showed the fastest learning. In contrast, most networks in other conditions either failed to recruit the directly connected version of a knowledge source or failed to develop weights that could make effective use of that recruitment. Nonetheless, all knowledge-laden conditions were superior in learning speed to the condition without any previous knowledge. It is difficult to pinpoint the nature of all of these advantages of knowledge over no knowledge.

===Insert Figure 13 about here===

4.3.2 Selection of Relevant Knowledge from Two Sources

In this experiment, networks first learned two tasks of varying relevance to the target task of learning a vertical rectangle (Rect90). The names of the various knowledge conditions, their relations to the target, and the mean times each network was recruited during input phases are shown in Table VI. The descriptions of the figures for each condition were provided in Table V. The two means in each row were compared with a *t*-test for paired samples. In each case, except the last, the mean difference was significant, $p < .001$, $df = 19$. Exact knowledge was preferred over relevant, irrelevant, or overly complex knowledge. Exact knowledge embedded within overly complex knowledge was also preferred over relevant, but inexact knowledge. The nominally irrelevant circle networks were recruited more often than relevant, inexact knowledge and as often as overly complex, but exact knowledge. Only one single-hidden-unit was recruited by the 120 networks in this experiment.

===Insert Table VI about here===

The preference for recruiting exact and exact but overly complex knowledge sources can be understood in the same terms used to analyze single source experiment on rotation. Directly connected interpretations of these exact knowledge sources are often recruited because they predict error very well, and once recruited they require very little further modification.

5. Finding and Using Component Knowledge

In this paradigm, we tested whether KBCC can find and combine source knowledge of components to learn a new, more complex target problem comprised of these components, and whether use of these knowledge components speeds learning. The main component in these tasks is a 0.5 x 1.5 rectangle. The target task is a cross (Cross90) formed by two superposed rectangles (Rect0 and Rect90). The transformation used to create variation in knowledge relevance is rotation. All figures are centered at (0, 0). The various source knowledge conditions are shown in Table VII. Descriptions of the various source knowledge components were provided in Table V.

===Insert Table VII about here===

A factorial ANOVA of the epochs to victory yielded a main effect of knowledge condition, $F(9, 190) = 50$, $p < .0001$. The mean epochs to victory, with standard deviation bars and homogeneous subsets, based on the *LSD* post hoc comparison method, are shown in Figure 14. Exact knowledge produced the fastest learning, followed by knowledge of the two exact target components, knowledge of one of the two exact target components, all other sorts of knowledge, and finally the control condition without any knowledge.

===Insert Figure 14 about here===

Number of hidden units recruited during the acquisition of source knowledge can be used as an index of complexity for single sources. These were subjected to a factorial ANOVA, yielding a main effect of knowledge condition, $F(6, 133) = 236$, $p < .0001$. The mean number of hidden units recruited in the source acquisition phase, with standard deviation bars and homogeneous subsets, based on the Scheffe method of post hoc comparison, are shown in Figure 15. Single rectangles proved to be the simplest, followed by the circle and the cross at 90 degrees, and finally the cross at 45 degrees.

===Insert Figure 15 about here===

Output activation diagrams are plotted in Figure 16 for a network learning the cross target by recruiting its two basic components, the vertical and horizontal rectangles. Figures 15a and 15b show the horizontal and vertical component sources, respectively. Figures 15c, 15d, and 15e show the target network at the end of the first, second, and third output phases, respectively. As illustrated in Figure 15c, when learning the cross, the best solution without any hidden units is to classify all patterns as being outside of the target class because the target class contains only 57 of the 225 training patterns. The third output phase was the final output phase for this network. This network recruited the horizontal source rectangle first, followed by the vertical source rectangle, each in their directly connected versions.

===Insert Figure 16 about here===

More generally, 81% of the source networks recruited in this condition were directly connected versions. All of networks in the exact (un-rotated cross) knowledge condition recruited the directly connected source network, and 90% were finished after this single

recruitment. In these recruitments (whether of independent source components or exact source knowledge), the direct input-side connection weights were around 1 for the *x*-axis and *y*-axis input units and around 0 elsewhere. Again, the shapes of the target solutions closely resembled the shapes of the recruited source knowledge. Networks in conditions with rotated source components did not, in general, exhibit such straightforward behavior as did networks with these un-rotated components.

6. Summary of Learning Speed Ups

To compare the amount of learning speed up in the various experiments, we computed several indices of speed up. For each experiment, we divided mean epochs in the no-knowledge or irrelevant-knowledge condition by mean epochs in either the exact-knowledge or best-inexact-knowledge condition.[6] Because the results were different for scaling down to a small rectangle and scaling up to a large rectangle in the sizing experiment, the indices were computed separately for the two versions of that experiment. These indices of learning speed up, shown in Figure 17, range from 1.34 in the irrelevant/best-inexact measure in the rotation experiment to 15.51 in the none/exact index in the components experiment. In general, knowledge speeds up learning substantially everywhere, but there is considerable variation in the size of these effects. Figure 17 shows tendencies for exact knowledge to be more beneficial than inexact knowledge and for irrelevant knowledge to be more beneficial than no knowledge at all. Across indices and experiments, the overall mean speed-up factor is 5.29.

===Insert Figure 17 about here===

7. Generalization

Generalization tests with the 200 randomly determined test patterns are presented in Table VIII in terms of the mean and standard deviation of percent misclassification by KBCC networks in each experiment. The mean percent misclassification is 3%, and never exceeds 7% in any experiment, indicating good generalization performance.

===Insert Table VIII about here===

8. Discussion

8.1 Overview of Results

The present results show that KBCC is able to find, adapt, and use its existing knowledge in the learning of new problems, significantly shortening the learning time. When exact knowledge is present, it is recruited for a quick solution. The more relevant the knowledge is, the more likely it will be recruited for solution of a new problem and the faster that new learning is likely to be. If KBCC knows the components of its new task, then it recruits and combines those components into a solution, again with a significant speed up in learning. These are the sorts of qualities one would expect in a system that effectively uses its knowledge in new learning.

With a single source of knowledge in memory, KBCC tends to learn fastest with knowledge that matches the target exactly, followed by exact but embedded knowledge, close and relevant knowledge, distant relevant knowledge, irrelevant knowledge, and no knowledge at all. When learning a multi-component target task, KBCC learns faster with knowledge of both components than knowledge of only one component. With multiple sources of knowledge, there

[6] By "best" we mean fastest to learn.

is a tendency for KBCC to prefer to recruit sources in this same order, that is, to recruit the source that allows it to learn faster. Many of these results were traced to differential tendencies to recruit directly connected source networks. Such source networks are more likely to be recruited when the knowledge is exact or embedded, and is likely to speed learning.

Testing KBCC in the different domains of translation, sizing, and rotation provided evidence on the generality of our conclusions. Although these domains differ in their overall difficulty and in some aspects of the findings, there was considerable generality in the results. For example, in learning a small rectangle, recruiting a source rectangle sharpened the critical corners so that learning was fast and distance of relevant knowledge was irrelevant to learning speed. However, in the more difficult problem of scaling up to a larger rectangle, critical corners were smoothed by recruitment and had to be relearned, thus increasing learning time in relation to distance of the source knowledge. Thus, the patterns of results relating knowledge relevance to learning speed may be dampened or enhanced by various manipulations, but they are rarely reversed.

8.2 A Note on Irrelevant Source Knowledge

We had termed learning a circle as an *irrelevant* source knowledge because a circle lacks the critically important corners possessed by target and source rectangles. In single-source experiments, KBCC networks recruiting circular source knowledge were generally slower to learn rectangular targets than those recruiting rectangular source knowledge. However, recruitment of circular source knowledge was typically favored over, and was faster than, recruitment of single hidden units. In dual-source experiments, recruitment of circular source knowledge was less preferred than recruitment of exact knowledge, but sometimes more preferred (in translation and rotation experiments) and sometimes less preferred (in sizing experiments) than relevant knowledge. It is interesting to speculate about the utility of supposedly irrelevant circular source knowledge in some of these comparisons.

Single hidden units carve up a problem space with uniform hyper-planes, whereas candidate networks built on circular concepts employ a more complex geometry that may be similar to that of the target concept, thus raising correlation during recruitment phases and possibly lowering error during output phases. For example, both a rectangle and a circle separate inside patterns from outside patterns. When a different-shaped source is in the same region as the target, it may become a desirable object to recruit because it can correlate quite highly with target error. Such high correlations may lead to recruitment but the need to sharpen corners to meet the additional requirements of rectangular targets may prolong learning, leading to even more recruitments (cf. the multiple numbers of circles recruited in some conditions in Tables II, IV, and VI).

8.3 Relation to Previous Work

As noted earlier, the present work on KBCC bears some relation to previous neural network research on knowledge transfer, multitask learning, sequential learning, lifelong learning, input re-coding, knowledge insertion, and modularity.

Pratt (1993) developed a technique called discriminability-based transfer that uses the weights from a previously trained network to initialize a new network. This is probably the most obvious and straightforward idea for using knowledge in new learning. However, because it did

not work as well as expected, Pratt improved the technique by re-scaling the previous network's hyper-planes so that useful ones had large weights and less useful ones had small weights.

Caruana (1993, 1997) pioneered Multitask Learning (MTL) in which he trained a network on several tasks taken from the same domain in parallel, with a single output for each task. Such networks typically learned a common hidden-unit representation, which then proved useful for learning subsequent tasks from the same domain. Baxter (1995) proved that the number of examples required for learning any one task in an MTL paradigm decreases as a function of total number of tasks learned in parallel.

Silver and Mercer (1996) developed a method of sequential learning called task rehearsal. Here, old tasks are pseudo-rehearsed during new learning, generating patterns that can be added to the those of the target task. In pseudo-rehearsal, the network generates its own target vectors, using its current weights, rather than merely accepting them from the environment (Robins 1995). Separate learning rates for each task are used to control the impact of each source task, ensuring that the most related tasks have the most impact on learning.

Thrun and Mitchell (1993) engineered a technique they called lifelong learning, in which a network meta-learns the slope of the desired function at each training example. This is essentially the derivative of the function at an example output with respect to the input attribute vector. Then, in new learning, a meta-network makes slope predictions and estimates its accuracy for each new training example. This technique seems to rely not so much on knowledge representations as on search knowledge.

Clark and Thornton (1997) discussed the importance of networks being able to re-code their input in learning difficult, so-called Type-2 problems. Type-1 problems are those that can be solved by sampling the originally coded input data. In contrast, Type-2 problems need re-coding in order to use Type-1 knowledge. Ability to do this would require some degree of incremental learning, modularity, and perhaps representational re-description (Karmiloff-Smith 1992), but no specific algorithm was proposed.

Shavlik (1994) presented the KBANN algorithm for creating knowledge-based artificial neural networks. KBANN converts a set of symbolic rules embodying domain knowledge of a problem into a feed-forward neural network with the final rule conclusions as output units and intermediate rule conclusions as hidden units. Connection and bias weights are initialized to implement the conjunctive and disjunctive structures of the rules. Networks thus initialized with knowledge are then trained with examples to refine the knowledge. Training is typically faster than with standard networks with random weights and leads to better generalization. Following the training, symbolic rules can be extracted from the network.

Jordon and Jacobs (1994) proposed the Hierarchical Mixture of Experts (HME) architecture to decompose a problem into network modules. Distinct network modules become expert on subtasks, and cooperate on an overall solution using gating networks that learn to weight the modular expert contributions for the different parts of the problem. HME was found to learn the dynamics of a four-degree-of-freedom robot arm much faster than a multi-layer back-propagation network did.

8.4 Advantages of KBCC

In contrast to these previous methods for using knowledge in learning, KBCC uses established techniques from generative learning algorithms (Fahlman & Lebiere, 1990). KBCC

recruits existing networks as well as single units as it needs them in order to increase its computational power in the service of error reduction. Treating its existing networks like untrained single units, KBCC trains weights to the inputs of existing source networks to determine whether their outputs correlate with the target network's error. In addition, KBCC trains the output weights from a recruited network in order to incorporate it into a solution of the current problem. This process of adapting old knowledge to new task allows for further recruitment of either additional networks or single units. Indeed, the same network could be recruited more than once if this proved to be desirable for learning the target task. This adaptation of the outputs of the recruited source network allows KBCC to use knowledge that is only partly relevant to the new task. The ability to adjust both incoming and outgoing weights with respect to a source network gives KBCC considerable flexibility in its use of knowledge. The fact that units and sub-networks are installed in a cascade allows KBCC to build its new learning on top of any recruited knowledge.

KBCC may be one way of reducing a complex problem to a simpler, already known problem, as recommended by Clark and Thornton (1997). When a network is recruited, this effectively reinterprets a target problem as if it were an instance of a known problem. Further recruitments and output weight adjustments then craft this reinterpretation into a solution to the target problem.

During input phases, KBCC searches for the best linear transformation of the target network's inputs in relation to its source networks. This enables a potentially large range of input re-coding schemes. All of the linear transformations that were tried in the present simulations (translations, size-scaling, and rotation) produced satisfactory results in the sense of faster learning.

A direct comparison on translation problems showed that KBCC was considerably more effective than MTL in terms of speeding up learning (Shultz & Rivest, 2000b). In contrast to KBCC networks, MTL networks did not show any benefits of knowledge in terms of increased learning speed. MTL networks had particular difficulty extracting exact knowledge from an overly complex source network. Moreover, they often failed to learn their assigned source problem and thus had to be replaced before proceeding to the target phase. The primary reason that MTL does not speed the learning of new tasks is that it requires both old and new tasks to be freshly learned in parallel. In contrast, KBCC recruits its old knowledge without having to relearn it.

Unlike many of the previous techniques for which both the inputs and outputs of the source and target task must match precisely, KBCC can potentially recruit any sort of function to use in a new task. Source network inputs and outputs can be arranged in different orders, employ different coding methods, and exist in different numbers than those in the target network. Indeed, the only real constraint on what can be recruited by KBCC is that it must be possible to find the first derivative for a recruitment object. This need for the first derivative is due to the use of the quickprop algorithm for weight adjustment. If numerical estimation or an optimization algorithm that did not require first derivatives were used, even this restriction would disappear. This extreme flexibility means that functions created by means other than KBCC itself could be recruited. The wide range of recruitment objects would appear to offer considerably more power and flexibility than most knowledge-based learners provide.

When a source network is recruited by KBCC, it is thereafter treated as a black box module. Like the modules discussed by Fodor (1983), KBCC's recruited networks are computationally encapsulated sub-systems that interact with the rest of the system only through their inputs and outputs. Although Fodor (1983) proposed that such modules are innate and operate only in particular specialized areas such as perception and language processing, it is now recognized that modules can be learned and also operate within central cognition (Karmiloff-Smith 1992).

In contrast to larger and more homogeneous networks, modular neural networks restrict complexity to be proportional to problem size, easily incorporate prior knowledge, generalize effectively, learn multiple tasks, perform robustly, and are easily extended or modified (Gallinari 1995). The solutions of modular networks should also be easier to analyze than the solutions of homogeneous networks. Whatever recruited KBCC modules do with their input would not change from the time of their initial acquisition, although it still might be challenging to determine their role in the overall solution reached by the target network. Unlike the HME approach to modularity, the sub-networks in KBCC are gradually constructed through automatic learning rather than being designed ahead of time and being simultaneously present throughout the whole of learning.

KBCC also implements a natural resistance to the retroactive interference that often plagues sequential learning in neural networks (French 1992). Because each source network is an unchanged module, it never loses its original functionality, no matter how many times and ways that it is recruited. There is also no need to relearn old tasks while learning new ones as in Silver and Mercer's (1996) task rehearsal method and in Caruana’s (1993, 1997) MTL.

KBCC allows for a combination of learning by either analogy and/or induction. KBCC learns by analogy to its current knowledge whenever it can and switches to a more inductive mode if it needs to. Recruiting a network is learning by analogy, whereas recruiting a single unit is learning by induction. Both processes are seamlessly integrated in KBCC's approach to a new target task.

8.5 Future Work

In this paper, we assessed the speed up in learning that comes from recruiting existing relevant knowledge in the KBCC algorithm. KBCC should also be assessed for the possibility that it could learn a problem more deeply and generalize more effectively by virtue of recruiting such knowledge. Deeper learning and more effective generalization was not apparent in the current work because learning was allowed to proceed to completion in every condition.[7] Even without any stored knowledge, KBCC, which is then essentially equivalent to ordinary CC, is powerful enough to learn these non-linear problems by recruiting individual hidden units as needed. Assessing the impact of knowledge on the quality of learning would require impoverished training sets and/or assessments earlier in learning. This issue is the focus of work that is currently underway in our laboratory.

KBCC has so far been applied to only toy demonstration problems, albeit problems that might pose some difficulty for other learning algorithms. Use of these well understood and easy

[7] This is not to say that KBCC does not generalize well. The generalization results in Table VIII show that mean misclassification error on test problems was only 3%. The point made here is that there is not yet any demonstration that recruiting existing knowledge by KBCC networks improves depth of learning, as indexed by superior generalization.

to visualize problems enables us to explore the properties of KBCC in some detail and with growing confidence that the algorithm is working appropriately. Nonetheless, it would be interesting and important to try KBCC on real and even more difficult problems and with a larger and more realistic array of source networks. A number of realistic problems have already been the focus of work on knowledge transfer in neural networks. These include problems in speech recognition, medical diagnosis, DNA pattern discovery, and chess (Pratt 1993; Silver & Mercer, 1996). In addition to exploring such realistic problems, we also plan to apply KBCC to the simulation of psychological data on the use of knowledge in learning.

A significant difficulty could be anticipated as a KBCC system accumulates extensive experience. The problem is that searching an extensive knowledge base of source networks in input (recruitment) phases would become prohibitively expensive computationally. It seems reasonable in such circumstances to focus that search on source networks that could be expected to be particularly useful. Focusing these recruitment searches could perhaps be accomplished with weight-implemented heuristics such as similarity in inputs and outputs, recency of learning, and externally provided hints about what existing knowledge might be useful. In addition and perhaps more importantly, past recruitment of particular source networks for particular problems might be used to develop a task semantics that could further constrain these searches.

The current version of KBCC is able to find and use its existing knowledge in learning new tasks. This holds the promise of being able to undertake more realistic implementations of the kind of knowledge-based learning at which people excel.

Author Note

This research was supported by a grant from the Natural Sciences and Engineering Research Council of Canada. This work has benefited from comments from David Buckingham, Jacques Katz, Sylvain Sirois, Yoshio Takane, and two anonymous reviewers.

References

Baxter, J. (1995) Learning internal representations. Proceedings of the Eighth International Conference on Computational Learning Theory. Santa Cruz, CA: ACM Press.

Buckingham, D. & Shultz, T. R. (1994) A connectionist model of the development of velocity, time, and distance concepts. Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Erlbaum.

Caruana, R. (1993) Multitask learning: A knowledge-based source of inductive bias. Proceedings of the Tenth International Machine Learning Conference. San Mateo, CA: Morgan Kaufmann.

Caruana, R. (1997) Multitask learning. Machine Learning, 28, 41-75.

Clark, A. & Thornton, C. (1997) Trading spaces: Computation, representation, and the limits of uninformed learning. Behavioral and Brain Sciences, 20, 57-97.

Fahlman, S. E. (1988) Faster-learning variations on back-propagation: An empirical study. In D. S. Touretzky, G. E. Hinton, & T. J. Sejnowski (Eds.), Proceedings of the 1988 Connectionist Models Summer School. Los Altos, CA: Morgan Kaufmann.

Fahlman, S. E. & Lebiere, C. (1990) The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), Advances in Neural Information Processing Systems 2. Los Altos, CA: Morgan Kaufmann.

Fodor, J. A. (1983) The Modularity of Mind. Cambridge, MA: MIT Press.

French, R. (1992) Semi-distributed representations and catastrophic forgetting in connectionist networks. Connection Science, 4, 365–377.

Gallinari, P. (1995) Training of modular neural net systems. In M. A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks. Cambridge, MA: MIT Press.

Heit, E. (1994) Models of the effects of prior knowledge on category learning. Journal of Experimental Psychology: Learning, Memory, and Cognition, 20, 1264-1282.

Jordan, M. I. & Jacobs, R. A. (1994) Hierarchical mixtures of experts and the EM algorithm. Neural Computation, 6, 181-214.

Karmiloff-Smith, A. (1992) Beyond modularity: A Developmental Perspective on Cognitive Science. Cambridge, MA: MIT Press.

Keil, F. C. (1987) Conceptual development and category structure. In U. Neisser (Ed.), Concepts and Conceptual Development: Ecological and Intellectual Factors in Categorization. Cambridge: Cambridge University Press.

Mareschal, D. & Shultz, T. R. (1999) Development of children's seriation: A connectionist approach. Connection Science, 11, 149-186.

Murphy, G. L. (1993) A rational theory of concepts. The Psychology of Learning and Motivation, 29, 327-359.

Nakamura, G. (1985) Knowledge-based classification of ill-defined categories. Memory and Cognition, 13, 377-384.

Pazzani, M. J. (1991) Influence of prior knowledge on concept acquisition: Experimental and computational results. Journal of Experimental Psychology: Learning, Memory, and Cognition, 17, 416-432.

Pratt, L. Y. (1993) Discriminability-based transfer between neural networks. Advances in Neural Information Processing Systems 5. San Mateo, CA: Morgan Kaufmann.

Robins, A. V. (1995) Catastrophic forgetting, rehearsal, and pseudorehearsal. Connection Science, 7, 123-146.

Shavlik, J. W. (1994) A framework for combining symbolic and neural learning. Machine Learning, 14, 321-331.

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

Shultz, T. R., Buckingham, D., & Oshima-Takane, Y. (1994) A connectionist model of the learning of personal pronouns in English. In S. J. Hanson, T. Petsche, M. Kearns, & R. L. Rivest (Eds.), Computational Learning Theory and Natural Learning Systems, Vol. 2: Intersection between Theory and Experiment. Cambridge, MA: MIT Press.

Shultz, T. R., Mareschal, D., & Schmidt, W. C. (1994) Modeling cognitive development on balance scale phenomena. Machine Learning, 16, 57-86.

Shultz, T. R., & Rivest, F. (2000a) Knowledge-based cascade-correlation. Proceedings of the International Joint Conference on Neural Networks. Los Alamitos, CA: IEEE Computer Society Press.

Shultz, T. R., & Rivest, F. (2000b) Using knowledge to speed learning: A comparison of knowledge-based cascade-correlation and multi-task learning. Proceedings of the Seventeenth International Conference on Machine Learning. San Francisco: Morgan Kaufmann.

Silver, D. & Mercer, R. (1996) The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. Connection Science, 8, 277-294.

Sirois, S. & Shultz, T. R. (1998) Neural network modeling of developmental effects in discrimination shifts. Journal of Experimental Child Psychology, 71, 235-274.

Thrun, S. & Mitchell, T. (1993) Integrating inductive neural network learning and explanation-based learning. In R. Bajcsy (Ed.), Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. San Mateo, CA: Morgan Kaufmann.

Wisniewski, E. J. (1995) Prior knowledge and functionally relevant features in concept learning. Journal of Experimental Psychology: Learning, Memory, and Cognition, 21, 449-468.

Table I

Single-source Knowledge Conditions for Translation Experiments

Name	Description	Relation to target
RectL	Rectangle centered at (-4/7, 0)	Exact
2RectLC	2 rectangles, centered at (-4/7, 0) and (0, 0)	Exact/near, overly complex
RectC	Rectangle centered at (0, 0)	Near relevant
RectR	Rectangle centered at (4/7, 0)	Far relevant
2RectCR	2 rectangles, centered at (0, 0) and (4/7, 0)	Near/far, overly complex
Circle	Circle centered at (0, 0) with radius 0.5	Irrelevant
None	No knowledge	None

Table II

Dual-source Knowledge Conditions and Mean Networks Recruited in Translation Experiments

Name	Relation to target	Mean networks recruited			
		RectL	RectR	2RectLC	Circle
RectL, RectR	Exact vs. Relevant	1.0	0.0	n/a	n/a
RectL, 2RectLC	Exact vs. Overly complex	0.95	n/a	0.15	n/a
RectL, Circle	Exact vs. Irrelevant	1.05	n/a	n/a	0.0
RectR, 2RectLC	Relevant vs. Overly complex	n/a	0.15	1.25	n/a
RectR, Circle	Relevant vs. Irrelevant	n/a	1.25	n/a	2.55
2RectLC, Circle	Overly complex vs. Irrelevant	n/a	n/a	1.45	0.15

Table III

Single-source Knowledge Conditions for Sizing Experiments

Name	Description	Relation to target RectL	Relation to target RectS
RectS	Rectangle of width 6/14	Far relevant	Exact
RectM	Rectangle of width 14/14	Near relevant	Near relevant
RectL	Rectangle of width 22/14	Exact	Far relevant
Circle	Center at (0, 0), radius 0.5	Irrelevant	Irrelevant
None	No knowledge	None	None

Table IV

Dual-source Knowledge Conditions and Mean Networks Recruited for Sizing Experiments

Name	Relation to target	Mean networks recruited			*t*(19)	*p* <
		RectS	RectL	Circle		
Target: RectS						
RectS, RectL	Exact vs. Relevant	1.05	0.60	n/a	3.33	.005
RectS, Circle	Exact vs. Irrelevant	1.00	n/a	0.45	3.58	.005
RectL, Circle	Relevant vs. Irrelevant	n/a	1.50	0.45	4.70	.001
Target: RectL						
RectL, RectS	Exact vs. Relevant	0.15	1.20	n/a	11.92	.001
RectL, Circle	Exact vs. Irrelevant	n/a	1.05	0.0	21.00	.001
RectS, Circle	Relevant vs. Irrelevant	2.65	n/a	3.40	1.25	ns

Table V

Single-source Knowledge Conditions for Rotation Experiments

Name	Description	Relation to target
Rect90	Vertical rectangle	Exact
Rect45	Diagonal rectangle	Near relevant
Rect0	Horizontal rectangle	Far relevant
Cross90	2 superposed rectangles	Exact/far, overly complex
Cross45	2 superposed rectangle forming diagonal cross	Near, overly complex
Circle	Circle centered at (0, 0) with radius 0.5	Irrelevant
None	No knowledge	None

Table VI

Dual-source Knowledge Conditions and Mean Networks Recruited for Rotation Experiments

Name	Relation to target	Mean networks recruited			
		Rect90	Rect0	Cross90	Circle
Rect90, Rect0	Exact vs. Relevant	1.35	0.25	n/a	n/a
Rect90, Cross90	Exact vs. Overly complex	1.00	n/a	0.35	n/a
Rect90, Circle	Exact vs. Irrelevant	1.25	n/a	n/a	0.25
Rect0, Cross90	Relevant vs. Overly complex	n/a	0.35	2.05	n/a
Rect0, Circle	Relevant vs. Irrelevant	n/a	.60	n/a	2.00
Cross90, Circle	Overly complex vs. Irrelevant	n/a	n/a	1.25	1.15

Table VII

Source Knowledge Conditions for Component Experiments

Name	Relation to target
Cross90	Exact
Rect0 & Rect90	Both components
Rect0	1st component
Rect90	2nd component
Cross45	Rotated 45 degrees
Rect45 & Rect135	Both components rotated
Rect45	1st rotated component
Rect135	2nd rotated component
Circle	Irrelevant
None	No knowledge

Table VIII

Mean and Standard Deviation of Percent Misclassification Error on Test Problems

Experiment	Mean	SD
Rotation/single	.027	.009
Rotation/dual	.025	.010
Translation/single	.030	.013
Translation/ dual	.023	.012
Scale-up/single	.040	.012
Scale-down/single	.031	.015
Scale-up/dual	.035	.018
Scale-down/dual	.020	.011
Cross from components	.070	.018

Figure Captions

Figure 1. A KBCC network with two hidden units, the first of which is a previously learned sub-network and the second a single unit. The network is shown in the third output phase. Dashed lines represent trainable weights, and solid lines represent frozen weights. Thin lines represent single weights; thick lines represent vectors of weights entering and exiting the recruited sub-network, which may have multiple inputs and multiple outputs.

Figure 2. Connection scheme for a sample KBCC network with two inputs, two outputs, one recruited network, and a recruited hidden unit.

Figure 3. Mean epochs to victory in the target phase of the translation experiment, with standard deviation bars and homogeneous subsets.

Figure 4. Output activation diagrams showing exact but overly complex source knowledge (a), the target network at the end of the first output phase (b), and the target solution at the end of the second output phase (c) after recruiting the source knowledge in a.

Figure 5. Mean epochs to victory in the target phase of the small rectangle condition of the sizing experiment, with standard deviation bars and homogeneous subsets.

Figure 6. Mean epochs to victory in the target phase of the large rectangle condition of the sizing experiment, with standard deviation bars and homogeneous subsets.

Figure 7. Output activation diagrams for a network learning a small rectangle, showing near relevant source knowledge (a) and the final target solution at the end of the second output phase (b) after recruiting the knowledge in a.

Figure 8. Output activation diagrams for a network learning a small rectangle, showing far relevant source knowledge (a) and the final target solution at the end of the second output phase (b) after recruiting the knowledge in a.

Figure 9. Output activation diagrams for a network learning a large rectangle, showing near relevant source knowledge (a) and target solutions at the end of the second (b), third (c), and fourth and final (d) output phases.

Figure 10. Output activation diagrams for a network learning a large rectangle, showing far relevant source knowledge (a) and target solutions at the end of the second (b), fifth (c), and sixth and final (d) output phases.

Figure 11. Mean input connection weights for *x*-axis input units learned in the sizing experiment while recruiting a directly connected source network.

Figure 12. Mean epochs to victory in the target phase of the rotation experiment, with standard deviation bars and homogeneous subsets.

Figure 13. Output activation diagrams for a network learning a vertical rectangle after recruiting directly connected exact but overly complex source knowledge (a). Target solutions are shown at the end of the second (b) and third and final (c) output phases.

Figure 14. Mean epochs to victory in the target phase of the components experiment, with standard deviation bars and homogeneous subsets.

Figure 15. Mean hidden units recruited in the source-acquisition phase of the components experiment, with standard deviation bars and homogeneous subsets.

Figure 16. Output activation diagrams for a network learning a cross target by recruiting its two components (a and b). Target solutions are shown at the end of the first (c), second (d), and third (final, e) output phases.

Figure 17. Speed-up factors in the various experiments in terms of mean epochs in the no-knowledge or irrelevant-knowledge condition divided by mean epochs in either the exact-knowledge or best-inexact-knowledge condition.

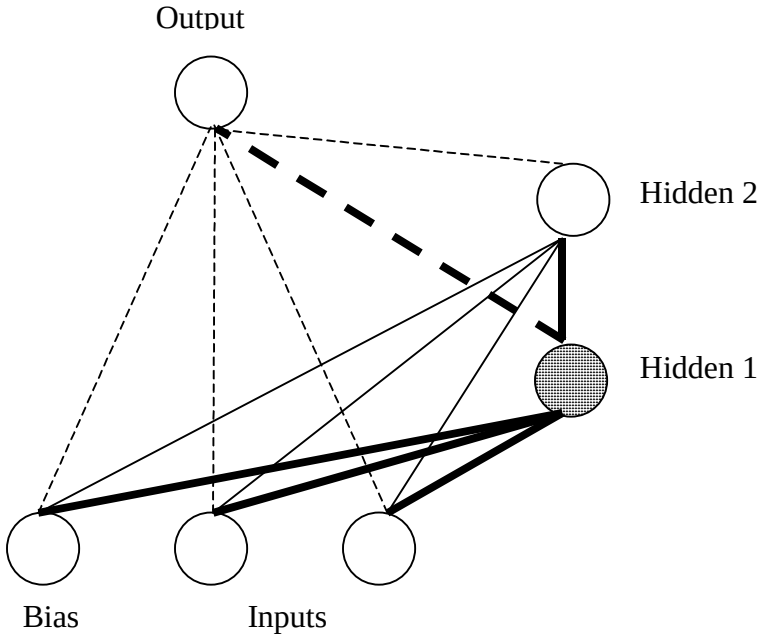


Figure 1. A KBCC network with two hidden units, the first of which is a previously learned sub-network and the second a single unit. The network is shown in the third output phase. Dashed lines represent trainable weights, and solid lines represent frozen weights. Thin lines represent single weights; thick lines represent vectors of weights entering and exiting the recruited sub-network, which may have multiple inputs and multiple outputs.

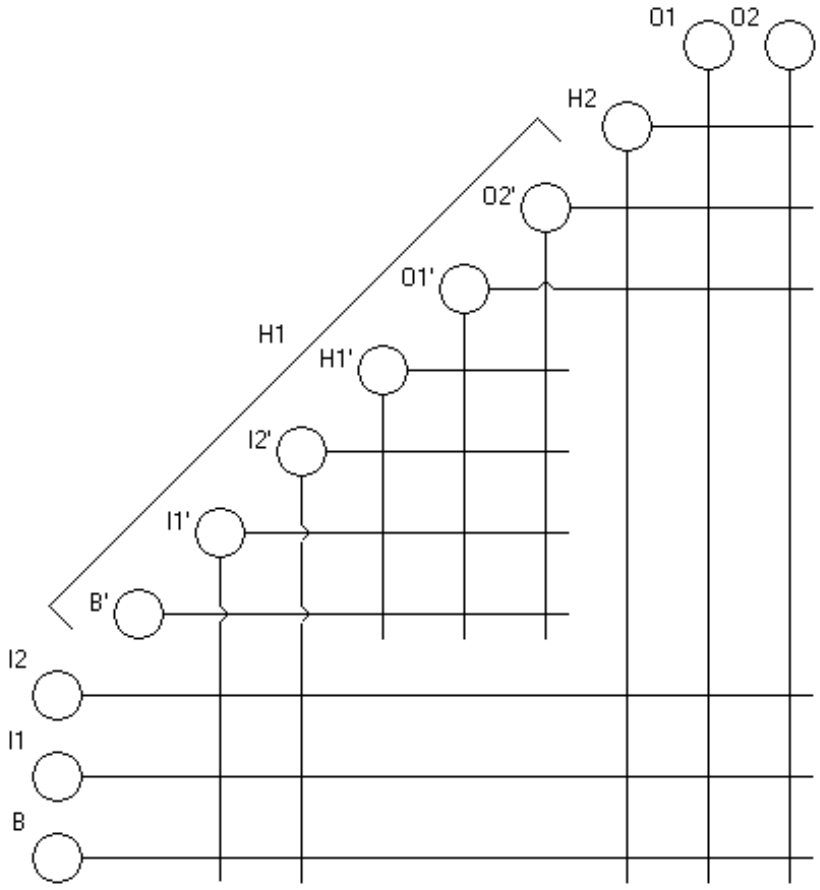


Figure 2. Connection scheme for a sample KBCC network with two inputs, two outputs, one recruited network, and a recruited hidden unit.

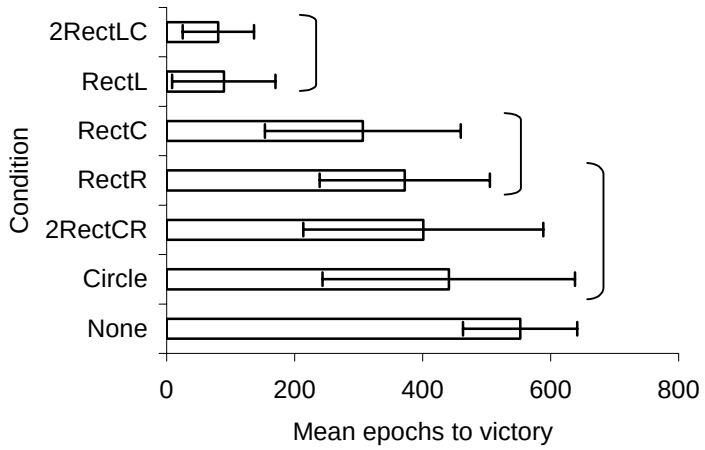
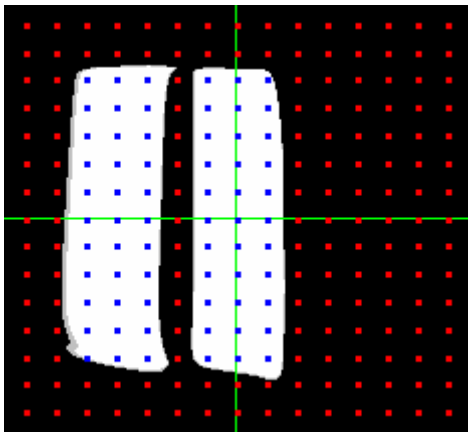
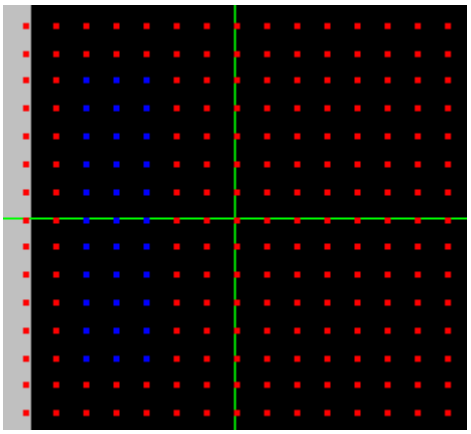


Figure 3. Mean epochs to victory in the target phase of the translation experiment, with standard deviation bars and homogeneous subsets.

a.



b.



c.

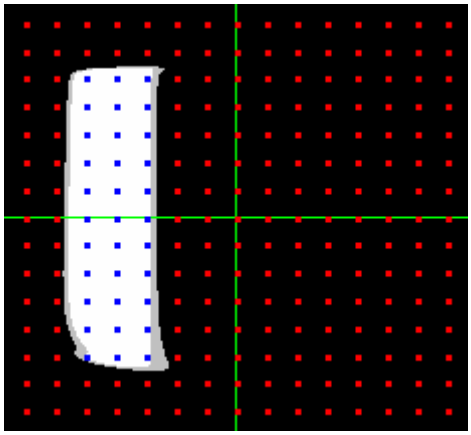


Figure 4. Output activation diagrams showing exact but overly complex source knowledge (a), the target network at the end of the first output phase (b), and the target solution at the end of the second output phase (c) after recruiting the source knowledge in a.

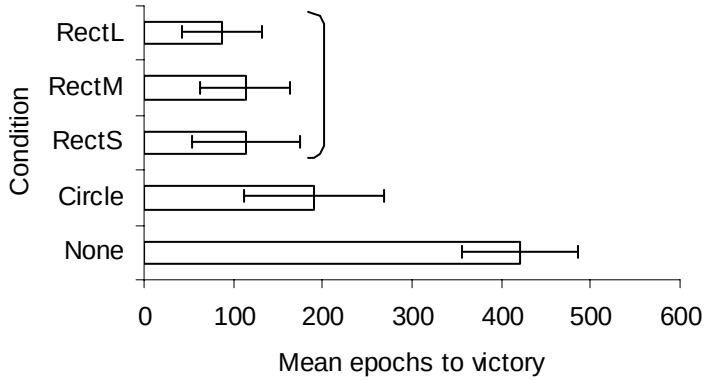


Figure 5. Mean epochs to victory in the target phase of the small rectangle condition of the sizing experiment, with standard deviation bars and homogeneous subsets.

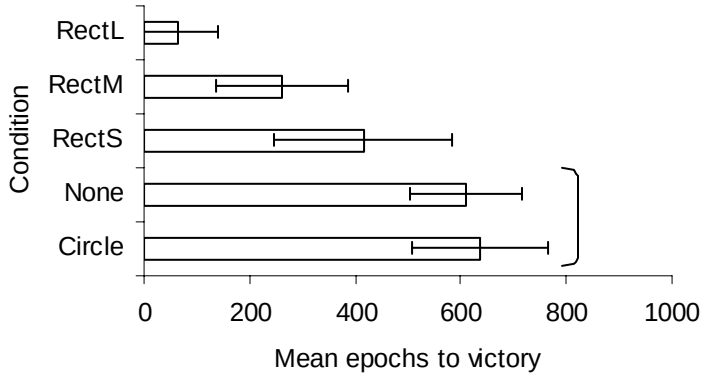


Figure 6. Mean epochs to victory in the target phase of the large rectangle condition of the sizing experiment, with standard deviation bars and homogeneous subsets.

a. b.

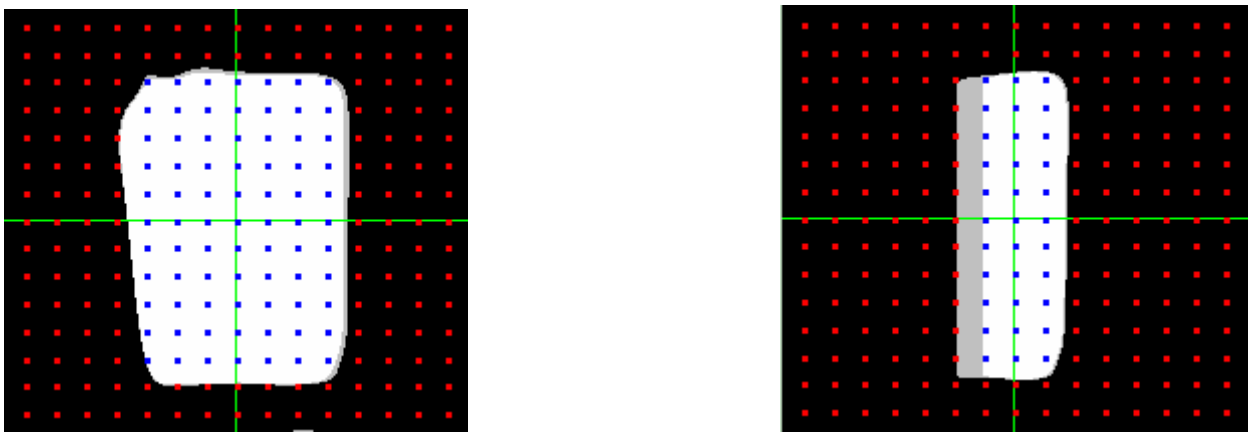


Figure 7. Output activation diagrams for a network learning a small rectangle, showing near relevant source knowledge (a) and the final target solution at the end of the second output phase (b) after recruiting the knowledge in a.

a. b.

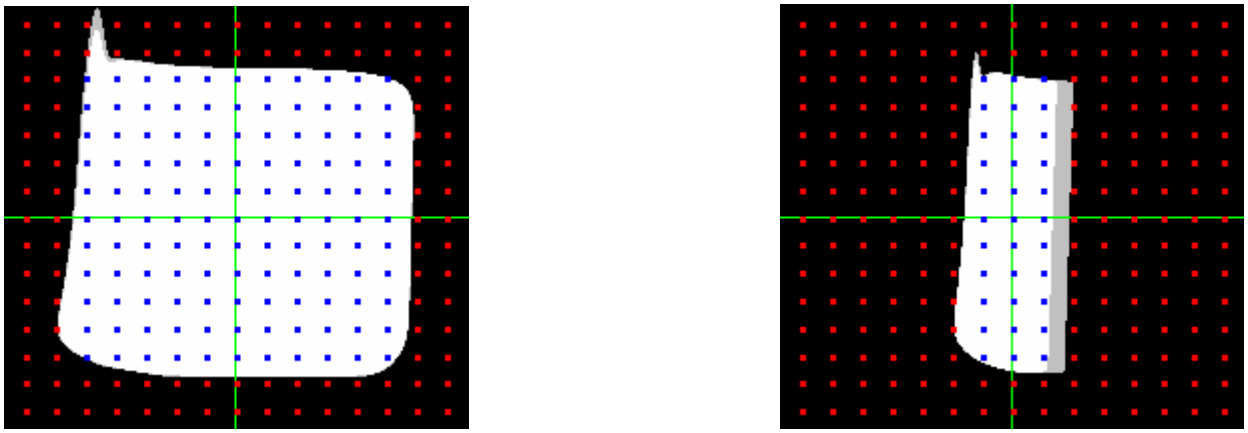
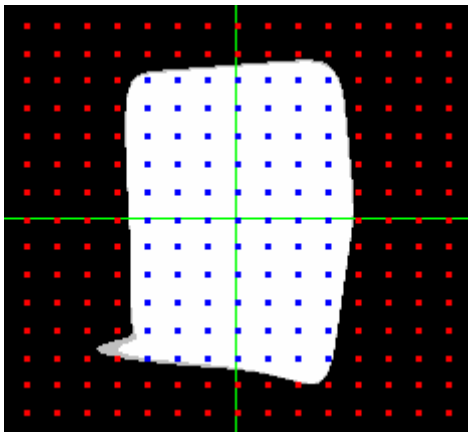
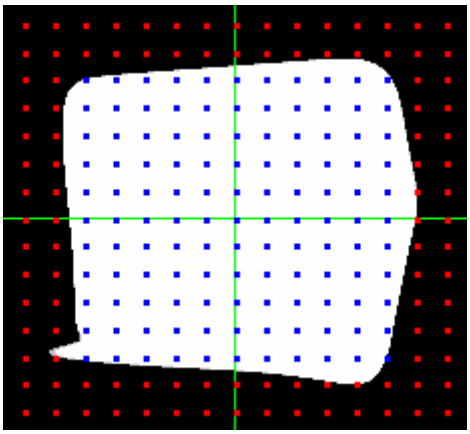


Figure 8. Output activation diagrams for a network learning a small rectangle, showing far relevant source knowledge (a) and the final target solution at the end of the second output phase (b) after recruiting the knowledge in a.

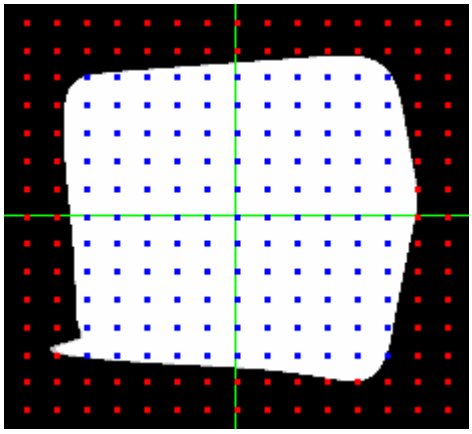
a.



b.



c.



d.

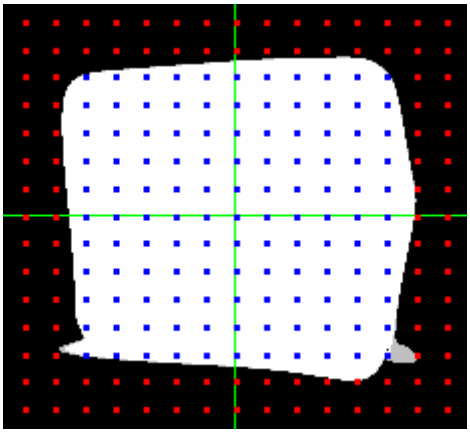
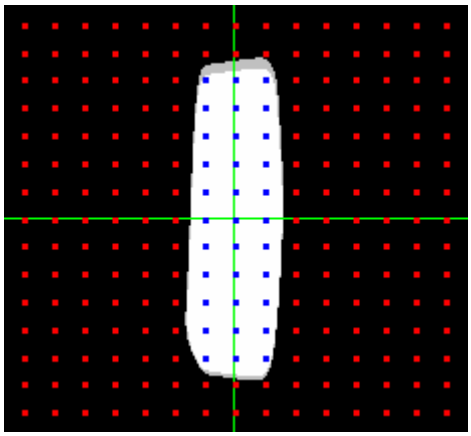
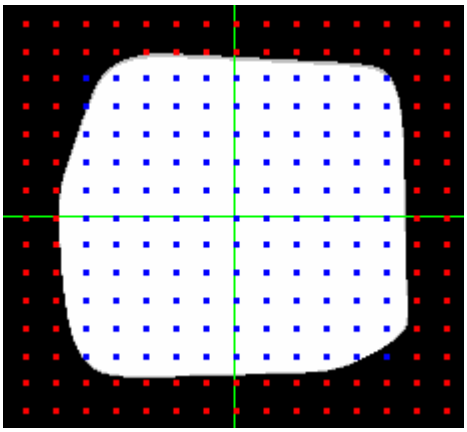


Figure 9. Output activation diagrams for a network learning a large rectangle, showing near relevant source knowledge (a) and target solutions at the end of the second (b), third (c), and fourth and final (d) output phases.

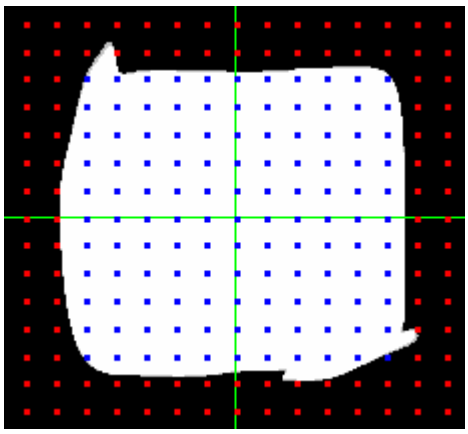
a.



b.



c.



d.

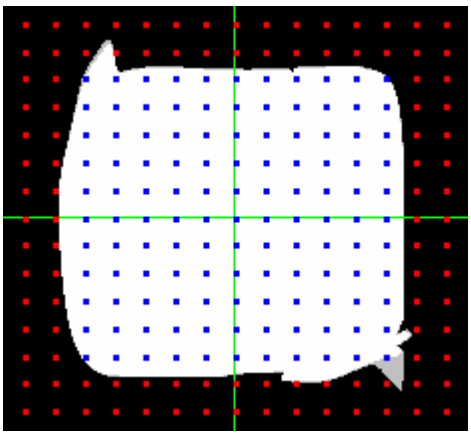


Figure 10. Output activation diagrams for a network learning a large rectangle, showing far relevant source knowledge (a) and target solutions at the end of the second (b), fifth (c), and sixth and final (d) output phases.

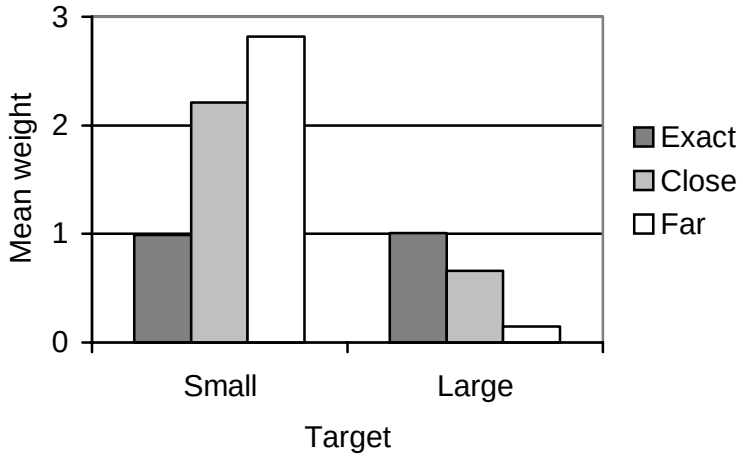


Figure 11. Mean input connection weights for *x*-axis input units learned in the sizing experiment while recruiting a directly connected source network.

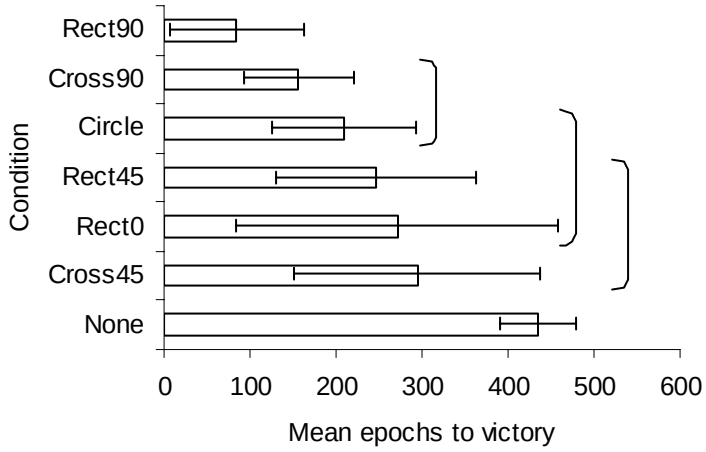
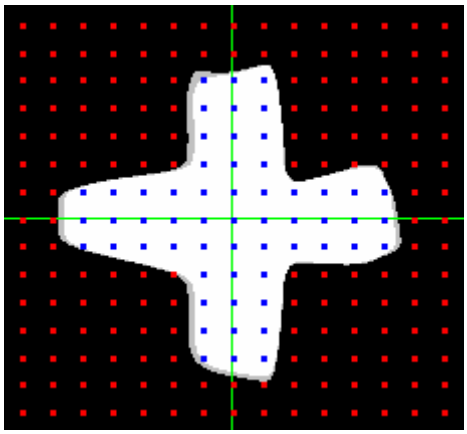
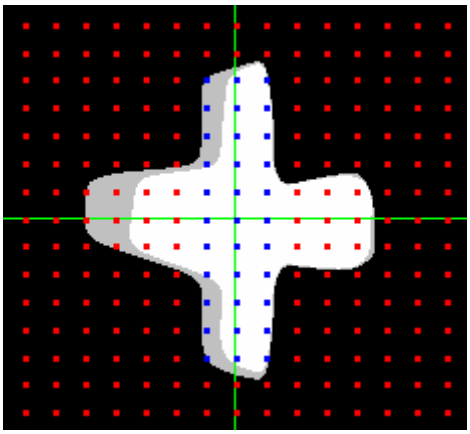


Figure 12. Mean epochs to victory in the target phase of the rotation experiment, with standard deviation bars and homogeneous subsets.

a.



b.



c.

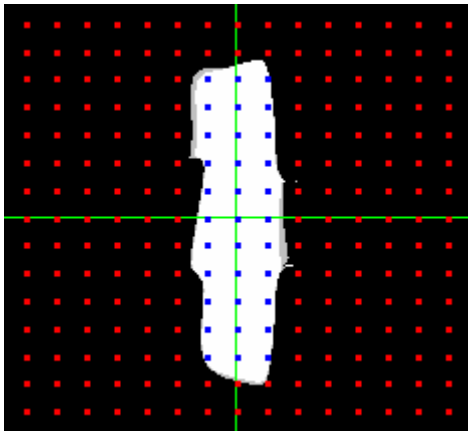


Figure 13. Output activation diagrams for a network learning a vertical rectangle after recruiting directly connected exact but overly complex source knowledge (a). Target solutions are shown at the end of the second (b) and third and final (c) output phases.

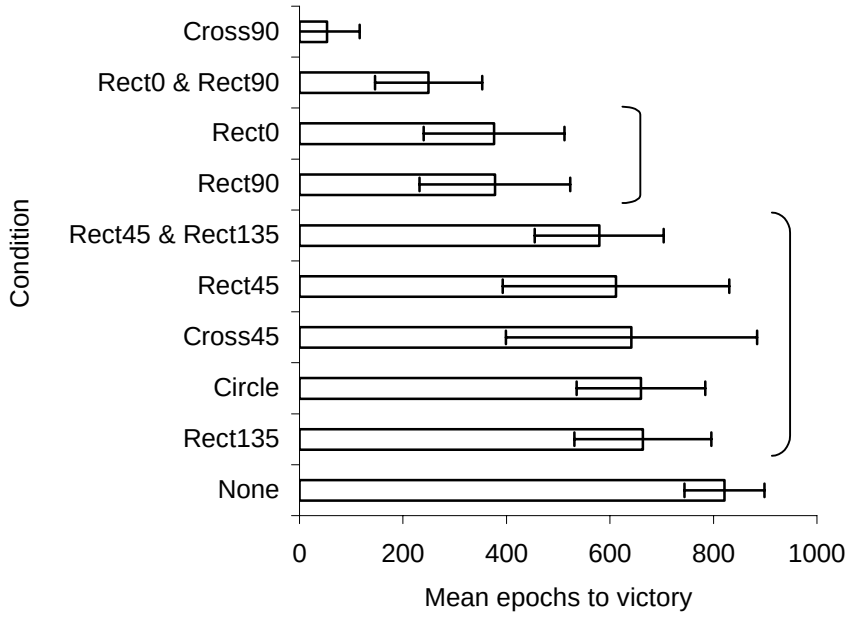


Figure 14. Mean epochs to victory in the target phase of the components experiment, with standard deviation bars and homogeneous subsets.

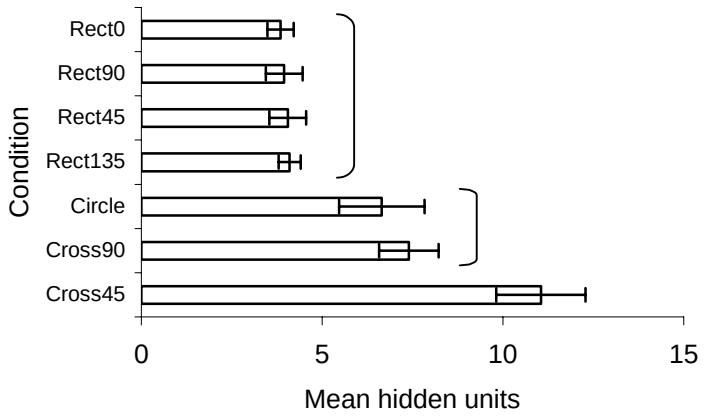


Figure 15. Mean hidden units recruited in the source-acquisition phase of the components experiment, with standard deviation bars and homogeneous subsets.

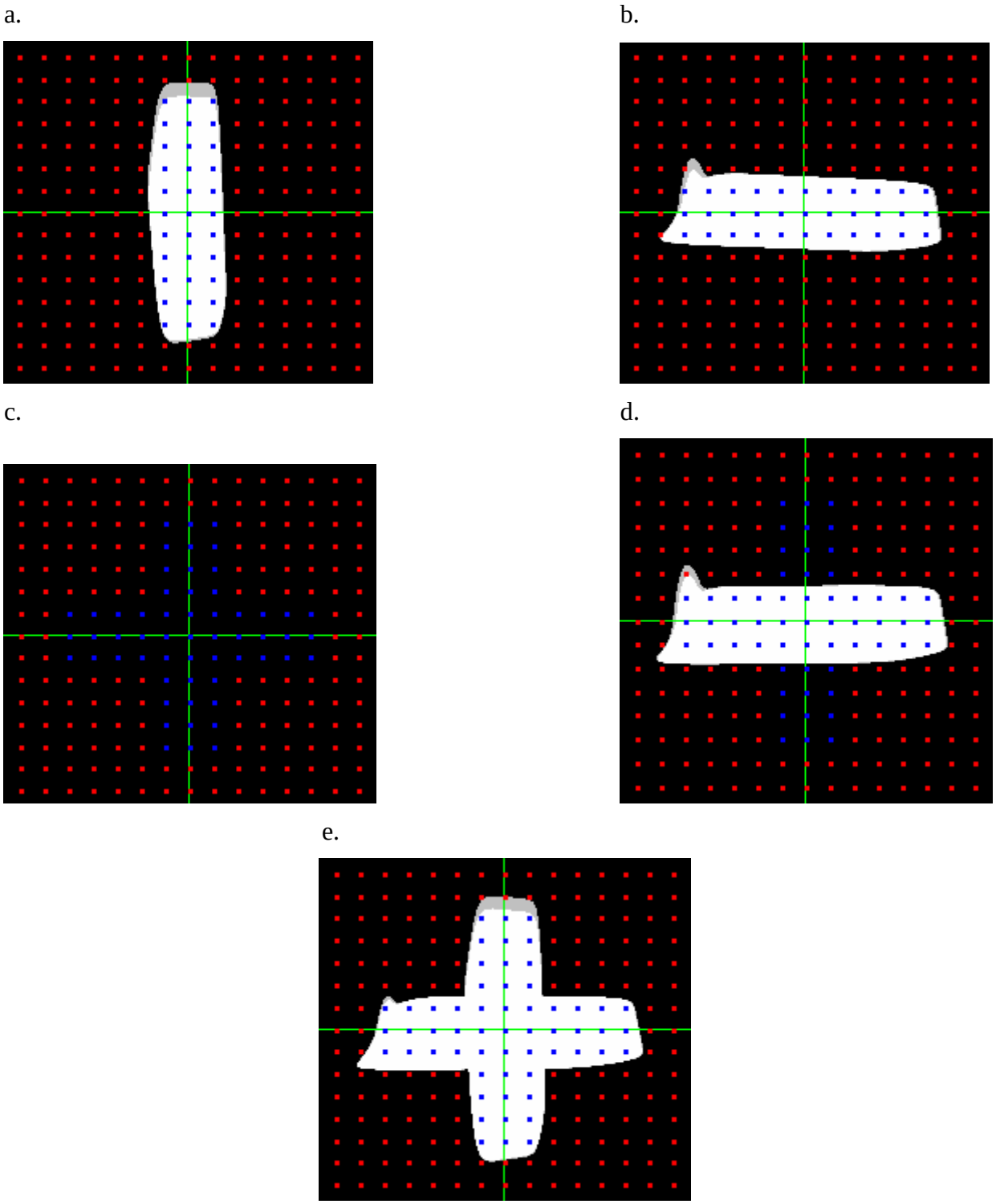


Figure 16. Output activation diagrams for a network learning a cross target by recruiting its two components (a and b). Target solutions are shown at the end of the first (c), second (d), and third (final, e) output phases.

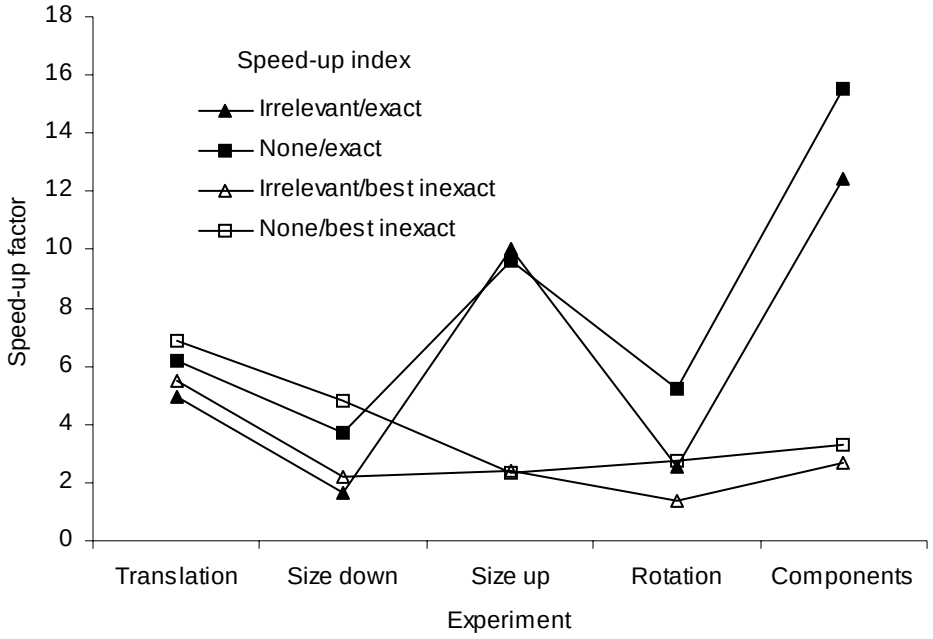


Figure 17. Speed-up factors in the various experiments in terms of mean epochs in the no-knowledge or irrelevant-knowledge condition divided by mean epochs in either the exact-knowledge or best-inexact-knowledge condition.